BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(1)

(19)日本国特許庁（ＪＰ）　(12)公表特許公報（Ａ）　(11)特許出願公表番号

特表平8－500200

(43)公表日　平成８年(1996)１月９日

(51)Int.Cl.[6]	識別記号	庁内整理番号	Ｆ Ｉ		
Ｇ０６Ｆ　17/21					
3/14	３５０　Ａ	7323－5B			
9/06	４１０　Ｓ	7230－5B			
		9288－5L	Ｇ０６Ｆ　15／20	５７０　Ｄ	
		9288－5L		５９６　Ｂ	

審査請求　有　　予備審査請求　有　（全126頁）　最終頁に続く

(21)出願番号　特願平6－514206
(86)(22)出願日　平成５年(1993)11月24日
(85)翻訳文提出日　平成７年(1995)５月30日
(86)国際出願番号　ＰＣＴ／ＵＳ９３／１１４６８
(87)国際公開番号　ＷＯ９４／１４１１５
(87)国際公開日　平成６年(1994)６月23日
(31)優先権主張番号　０７／９８４，８６８
(32)優先日　1992年12月１日
(33)優先権主張国　米国（ＵＳ）
(81)指定国　ＥＰ(ＡＴ，ＢＥ，ＣＨ，ＤＥ，ＤＫ，ＥＳ，ＦＲ，ＧＢ，ＧＲ，ＩＥ，ＩＴ，ＬＵ，ＭＣ，ＮＬ，ＰＴ，ＳＥ)，ＣＡ，ＪＰ

(71)出願人　マイクロソフト　コーポレイション
アメリカ合衆国　ワシントン州　98052－6399　レッドモンド　ワン　マイクロソフト　ウェイ（番地なし）
(72)発明者　コッポル　スリニヴァサ　アール
アメリカ合衆国　ワシントン州　98052　レッドモンド　トゥーハンドレッドアンドサーティシックスス　アベニュー　ノースイースト　2402
(74)代理人　弁理士　中村　稔　（外７名）

最終頁に続く

(54)【発明の名称】　埋め込まれたオブジェクトとイン・プレース対話するための方法及びシステム

(57)【要約】
コンテナーオブジェクト内に収容されたコンテニーオブジェクトと対話するためのコンピュータ方法及びシステムが提供される。本発明の好ましい実施例において、コンテナーオブジェクトは、該コンテナーオブジェクトと対話するためのコンテナーリソースを有するコンテナーウインドウ環境を伴うコンテナーアプリケーションを有している。コンテニーオブジェクトは、該コンテニーオブジェクトと対話するためのサーバリソースをもつサーバウインドウ環境を伴うサーバアプリケーションを有している。本発明の方法は、コンテナーウインドウ環境をディスプレイ装置に表示する。次いで、ユーザはコンテニーオブジェクトを選択する。コンテニーオブジェクトの選択に応答して、この方法は、複数のサーバリソースを、表示されたコンテナーウインドウ環境と一体化する。ユーザがサーバリソースを選択したときに、この方法は、サーバアプリケーションを呼び出して、サーバリソース選択を処理する。これに対し、ユーザがコンテナーリソースを選択したときは、この方法は、コンテナーアプリケーションを呼び出して、そのコンテナーリソー

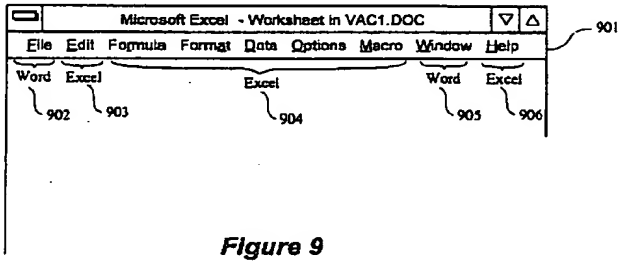

Figure 9

【特許請求の範囲】

１．コンピュータシステムにおいてコンテナーオブジェクト内に収容されたコンテニーオブジェクトをアクチベートするための方法であって、コンテナーオブジェクトは、コンテナーウインドウ環境を伴うコンテナーアプリケーションを有し、コンテナーウインドウ環境は、コンテナーオブジェクトと対話するためのコンテナーリソースを有し、コンテニーオブジェクトは、該コンテニーオブジェクトと対話するためのサーバリソースを伴うサーバアプリケーションを有し、上記方法は、

上記コンテナーウインドウ環境を表示し、

上記表示されたコンテナーウインドウ環境内に上記コンテニーオブジェクトを表示し、

上記コンテニーオブジェクトを選択し、そして

複数のサーバリソースを上記表示されたコンテナー環境と一体化して、ユーザがサーバリソースを選択するときに、サーバアプリケーションがサーバリソースの選択を処理するようにする、

という段階を備えたことを特徴とする方法。

２．上記コンテナーアプリケーションはコンテナーメニューを有し、上記サーバアプリケーションはサーバメニューを有し、複数のサーバリソースを一体化する上記段階は、サーバメニュー及びコンテナーメニューを有する複合メニューバーを発生する請求項１に記載の方法。

３．上記一体化段階は、サーバメニュー及びコンテナーメニューを複合メニューバーにおいてインターリーブする請求項２に記載の方法。

４．上記コンテナーアプリケーションは、複数のメニューを表示するためのメニューバーを有し、そして上記複合メニューバーは、コンテナーアプリケーションのメニューバーとして表示される請求項２に記載の方法。

５．コンピュータシステムにおいてコンテナーオブジェクト内に収容されたコンテニーオブジェクトと対話するための方法であって、コンテナーオブジェクトは、コンテナーウインドウ環境を伴うコンテナーアプリケーションを有し、コンテナーウインドウ環境は、コンテナーオブジェクトと対話するためのコンテナー

リソースを有し、コンテニーオブジェクトは、サーバウインドウ環境を伴うサーバアプリケーションを有し、サーバウインドウ環境は、コンテニーオブジェクトと対話するためのサーバリソースを有し、上記方法は、

上記コンテナーウインドウ環境を表示し、

上記コンテニーオブジェクトを選択し、

複数のサーバリソースを上記コンテナーウインドウ環境と一体化し、

ユーザがサーバリソースを選択するときには、そのサーバリソース選択を処理するようにサーバアプリケーションを呼び出し、そして

ユーザがコンテナーリソースを選択するときには、そのコンテナーリソース選択を処理するようにコンテナーアプリケーションを呼び出す、

という段階を備えたことを特徴とする方法。

６．上記コンテナーアプリケーションは、データを表示するためのウインドウを有し、そしてそのウインドウ内にサーバリソースを配置するようにコンテナーアプリケーションと交渉する段階を備えた請求項５に記載の方法。

７．サーバアプリケーションのみがサーバリソースの知識を有する請求項６に記載の方法。

８．上記コンテナーアプリケーション及びサーバアプリケーションは、個別のコンピュータプロセスとして実行される請求項５に記載の方法。

９．上記コンテニーオブジェクトを表示し、そしてその表示されたコンテニーオブジェクトをハイライト処理して、サーバリソースがユーザ選択に使用できることを指示する段階を備えた請求項５に記載の方法。

１０．上記コンテナアプリケーションは、メッセージを受け取って処理するためのコンテナーメッセージハンドラーを有し、上記サーバアプリケーションは、メッセージを受け取って処理するためのサーバメッセージハンドラーを有し、そして上記方法は、更に、上記コンテナーメッセージハンドラーを特殊なメッセージハンドラーと置き換える段階を備え、この特殊なメッセージハンドラーは、コンテナーリソース選択メッセージをコンテナーメッセージハンドラーへ送ると共に、サーバリソース選択メッセージをサーバメッセージハンドラーへ送るものである請求項５に記載の方法。

１１．上記コンテナーアプリケーションはウインドウを有し、上記サーバアプリケーションはウインドウを有し、そして更に、

上記サーバアプリケーションウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして指定し、

ユーザからメニューコマンドを受け取り、そして

上記メニューコマンドを受け取るのに応答して、

上記コンテナーアプリケーションウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして指定し、

メニューニューモニックを受け取り、そして

メニューニューモニックを受け取るのに応答して、サーバアプリケーションウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして再指定する、

という段階を含む請求項５に記載の方法。

１２．上記コンテナーアプリケーションのウインドウは、フレームウインドウである請求項１１に記載の方法。

１３．上記コンピュータシステムはキー入力のためのキーボードを有し、上記コンテナーアプリケーションは、コンテナーリソースを選択するための複数のアクセラレータキー組合せを有し、上記サーバアプリケーションは、サーバリソースを選択するための複数のアクセラレータキー組合せを有し、そして上記コンテナーアプリケーションは、アクセラレータキー組合せを受け取るときに、サーバアプリケーションを呼び出して、サーバリソースが選択されたかどうか決定する請求項５に記載の方法。

１４．コンピュータシステムにおいてコンテナーオブジェクト内に収容されたコンテニーオブジェクトと対話するための方法であって、コンテナーオブジェクトは、複数のコンテナーメニューを伴う関連するコンテナーアプリケーションを有し、このコンテナーアプリケーションは、メニューのリストを表示するための関連するメニューバーを有し、コンテニーオブジェクトは、複数のサーバメニューを伴う関連するサーバアプリケーションを有し、上記方法は、

コンテナーメニュー及びサーバメニューを含む複合メニューリストを発生し、

上記複合メニューリストをメニューバーに表示し、

表示されたコンテナーメニューをユーザが選択するのに応答して、コンテナーアプリケーションを呼び出してその選択されたメニューを処理し、そして

表示されたサーバメニューをユーザが選択するのに応答して、サーバアプリケーションを呼び出してその選択されたメニューを処理する、

という段階を備えたことを特徴とする方法。

１５．コンピュータシステムにおいて複数のアプリケーションからのメニューを一体化する方法であって、

各アプリケーションごとに１組のメニューグループを定義し、各メニューグループは複数のメニュー項目を有し、

上記メニューグループを複合メニューに結合し、

複合メニューをディスプレイ装置に表示し、

表示された複合メニューのメニューグループのメニュー項目を選択し、そして

その選択されたメニュー項目のメニューグループを定義するアプリケーションを呼び出す、

という段階を備えたことを特徴とする方法。

１６．上記複数のアプリケーションは、コンテナーアプリケーション及びサーバアプリケーションを含み、そして複合メニューを表示する上記段階は、複合メニューをコンテナーアプリケーションのメニューとして表示する請求項１５に記載の方法。

１７．コンピュータシステムにおいてウインドウハイアラーキを動的に結合する方法であって、上記コンピュータシステムは、ディスプレイ装置と、該ディスプレイ装置におけるウインドウの表示を管理するためのウインドウシステムとを有し、上記方法は、

第１のアプリケーションプログラムを呼び出しそして第１のウインドウハイアラーキを形成し、

上記第１のウインドウハイアラーキのウインドウを、第２のアプリケーションプログラムの親ウインドウとして指定し、そして　第２のアプリケーションプログラムを呼び出しそして第２のウインドウハイア

ラーキを形成し、ウインドウハイアラーキのルートウインドウが上記指定されたウインドウの子ウインドウとなるようにする、

という段階を備えたことを特徴とする方法。

１８．コンピュータシステムにおいてウインドウをスクロールする方法であって、上記コンピュータシステムは、ディスプレイ装置と、該ディスプレイ装置におけるウインドウの表示を管理するためのウインドウシステムとを有し、該ウインドウシステムは、ウインドウ内に表示されたデータをスクロールできるようにするものであり、上記方法は、

選択されるべきデータを指示し、該データは、外側のウインドウ内に収容された内側のウインドウ内に収容され、

外側のウインドウをスクロールするためのユーザ入力に応答して、上記指示されたデータを選択されたデータとして維持する、

という段階を備えたことを特徴とする方法。

１９．内側のウインドウがユーザの視野から外れてスクロールされたときに上記指示されたデータを選択されたものとして維持する段階を備えた請求項１８に記載の方法。

２０．内側のウインドウを、入力を受けるための入力フォーカスを有するものとして指定し、そして内側のウインドウがユーザの視野から外れてスクロールされたときに内側のウインドウを、入力フォーカスを有するものとして指定維持するという段階を備えた請求項１８に記載の方法。

２１．コンピュータシステムにおいてディスプレイ装置に表示されたオブジェクトが選択されたことを指示する方法であって、陰影付けされたパターンのボーダーをオブジェクトの周りに表示する段階を備えたことを特徴とする方法。

２２．コンテナーオブジェクト内に収容されたコンテニーオブジェクトをアクチベートするためのコンピュータシステムであって、上記コンテナーオブジェクトは、コンテナーウインドウ環境を伴うコンテナーアプリケーションを有し、該コンテナーウインドウ環境は、コンテナーオブジェクトと対話するためのコンテナーリソースを有し、コンテニーオブジェクトは、該コンテニーオブジェクトと対話するためのサーバリソースを伴うサーバアプリケーションを有し、上記シス

テムは、

上記コンテナーウインドウ環境を表示する手段と、

上記表示されたコンテナーウインドウ環境内に上記コンテニーオブジェクトを表示する手段と、

上記コンテニーオブジェクトを選択する手段と、

複数のサーバリソースを上記表示されたコンテナー環境と一体化する手段であって、ユーザがサーバリソースを選択するときに、サーバアプリケーションがサーバリソースの選択を処理するようにする手段と、

を備えたことを特徴とするコンピュータシステム。

２３．上記コンテナーアプリケーションはコンテナーメニューを有し、上記サーバアプリケーションはサーバメニューを有し、上記一体化手段は、サーバメニュー及びコンテナーメニューを有する複合メニューバーを発生する手段を備えた請求項２２に記載のシステム。

２４．上記一体化手段は、サーバメニュー及びコンテナーメニューを複合メニューバーにおいてインターリーブする手段を含む請求項２３に記載のシステム。

２５．上記コンテナーアプリケーションは、複数のメニューを表示するためのメニューバーを有し、そして上記複合メニューバーは、コンテナーアプリケーションのメニューバーとして表示される請求項２３に記載のシステム。

２６．コンテナーオブジェクト内に収容されたコンテニーオブジェクトと対話するためのコンピュータシステムであって、コンテナーオブジェクトは、コンテナーウインドウ環境を伴うコンテナーアプリケーションを有し、コンテナーウインドウ環境は、コンテナーオブジェクトと対話するためのコンテナーリソースを有し、コンテニーオブジェクトは、サーバウインドウ環境を伴うサーバアプリケーションを有し、サーバウインドウ環境は、コンテニーオブジェクトと対話するためのサーバリソースを有し、上記システムは、

上記コンテナーウインドウ環境をディスプレイ装置に表示する手段と、

上記コンテニーオブジェクトを選択する手段と、

コンテニーオブジェクトが選択されたときに複数のサーバリソースを上記コンテナーウインドウ環境と一体化する手段と、

サーバリソースを選択する手段と、

サーバリソースが選択されたときに、そのサーバリソース選択を処理するようにサーバアプリケーションを呼び出す手段と、

コンテナーリソースを選択する手段と、

コンテナーリソースが選択されたときに、そのコンテナーリソース選択を処理するようにコンテナーアプリケーションを呼び出す手段と、

を備えたことを特徴とするコンピュータシステム。

２７．上記コンテナーアプリケーションは、データを表示するためのウインドウを有し、そして上記一体化手段は、そのウインドウ内にサーバリソースを配置するようにコンテナーアプリケーションと交渉する手段を備えた請求項２６に記載のシステム。

２８．サーバアプリケーションのみがサーバリソースの知識を有する請求項２７に記載のシステム。

２９．上記コンテナーアプリケーション及びサーバアプリケーションは、個別のコンピュータプロセスとして実行される請求項２６に記載のシステム。

３０．上記コンテニーオブジェクトを表示する手段と、その表示されたコンテニーオブジェクトをハイライト処理して、サーバリソースがユーザ選択に使用できることを指示する手段とを備えた請求項２６に記載のシステム。

３１．上記コンテナアプリケーションは、メッセージを受け取って処理するためのコンテナーメッセージハンドラーを有し、上記サーバアプリケーションは、メッセージを受け取って処理するためのサーバメッセージハンドラーを有し、そして上記システムは、更に、上記コンテナーメッセージハンドラーを特殊なメッセージハンドラーと置き換える手段を備え、該特殊なメッセージハンドラーは、コンテナーリソース選択メッセージをコンテナーメッセージハンドラーへ送ると共に、サーバリソース選択メッセージをサーバメッセージハンドラーへ送るものである請求項２６に記載のシステム。

３２．上記コンテナーアプリケーションはウインドウを有し、上記サーバアプリケーションはウインドウを有し、そして更に、

上記サーバアプリケーションウインドウを、ユーザ入力を受け取るための入力

フォーカスを有するものとして指定する手段と、

ユーザからメニューコマンドを受け取る手段と、

メニューコマンドを受け取るのに応答して、上記コンテナーアプリケーションウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして指定する手段と、

メニューニューモニックを受け取る手段と、

メニューニューモニックを受け取るのに応答して、サーバアプリケーションウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして再指定する手段と、

を備えた請求項２６に記載のシステム。

３３．上記コンテナーアプリケーションのウインドウは、フレームウインドウである請求項３２に記載のシステム。

３４．上記コンピュータシステムはキー入力のためのキーボードを有し、上記コンテナーアプリケーションは、コンテナーリソースを選択するための複数のアクセラレータキー組合せを有し、上記サーバアプリケーションは、サーバリソースを選択するための複数のアクセラレータキー組合せを有し、そしてアクセラレータキー組合せを受け取るときに、サーバアプリケーションを呼び出して、サーバリソースが選択されたかどうか決定する手段を備えた請求項２６に記載のシステム。

【発明の詳細な説明】

埋め込まれたオブジェクトとイン・プレース対話するための方法及びシステム

発明の分野

本発明は、一般に、リンクされそして埋め込まれたオブジェクトと対話するためのコンピュータ方法及びシステムに係り、より詳細には、コンテナーアプリケーションのコンテクスト内に収容されたオブジェクトを編集し、さもなくば、それと対話するための方法及びシステムに係る。

先行技術の説明

現在の文書処理コンピュータシステムは、ユーザが複合文書を作成できるようにする。複合文書とは、種々のフォーマットの情報を含む文書である。例えば、複合文書は、テキストフォーマット、チャートフォーマット、数字フォーマット等のデータを含む。図１は、複合文書の一例である。この例において、複合文書１０１は、ある製造オブジェクトに対するレポートとして形成される。この複合文書１０１は、チャートフォーマットで表されたスケジュールデータ１０２と、スプレッドシートフォーマットで表された予算データ１０３と、テキストフォーマットで表された説明データ１０４とを含んでいる。典型的な公知のシステムでは、ユーザは、プロジェクトマネージメントコンピュータプログラムを用いてスケジュールデータ１０２を発生し、そしてスプレッドシートコンピュータプログラムを用いて予算データ１０３を発生する。このデータが発生された後に、ユーザは、複合文書１０１を形成し、説明データ１０４を入力し、そしてワードプロセスコンピュータプログラムを使用してスケジュールデータ１０２と予算データ１０３を結合する。

図２は、スケジュールデータ、予算データ及び説明データを複合文書へいかに組み込むかを示している。ユーザは、プロジェクトマネージメントプログラム２０１を使用してスケジュールデータを発生し、そしてそのデータをクリップボード２０３に記憶する。ユーザは、スプレッドシートプログラム２０４を使用して予算データを発生し、次いで、そのデータをクリップボード２０３に記憶する。クリップボード２０３は、通常はいかなるプログラムによってもアクセスできる

記憶エリア（ディスク又はメモリ）である。プロジェクトマネージメントプログラム２０１及びスプレッドシートプログラム２０４は、通常は、データをプレゼンテーションフォーマットでクリップボードに記憶する。プレゼンテーションフォーマットとは、データが出力装置に容易に表示されるフォーマットである。例えば、プレゼンテーションフォーマットは、標準的なビットマップブロック転送動作（ＢｉｔＢｌｔ）で表示することのできるビットマップである。データをクリップボードに記憶することを、クリップボードに「コピーする」と称する。

データがクリップボード２０３にコピーされた後に、ユーザは、ワード処理プログラム２０６を始動して複合文書１０１を作成する。ユーザは、説明データ１０４を入力し、そしてクリップボード２０３にあるスケジュールデータ及び予算データをコピーすべき複合文書１０１内の位置を指定する。クリップボードから文書へデータをコピーすることを、クリップボードから「ペーストする」と称する。次いで、ワードプロセスプログラム２０６は、スケジュールデータ１０２及び予算データ１０３をクリップボード２０３から複合文書１０１の指定の位置へコピーする。クリップボードから複合文書へコピーされるデータは、「埋め込まれる」データと称する。ワードプロセスプログラム２０６は、この埋め込まれるデータを、複合文書１０１を出力装置においてレンダリングするときにＢｉｔＢｌｔ動作で表示する単純なビットマップとして処理する。ある公知システムにおいては、クリップボードは、一度に１つのコピーコマンドに対してデータを記憶することしかできない。このようなシステムでは、スケジュールデータは、クリップボードにコピーし、そして複合文書へペーストすることができる。次いで、予算データをクリップボードにコピーし、そして複合文書へペーストすることができる。

ワードプロセッサは、通常、テキストデータしか処理しないので、ワードプロセスプログラムのユーザは、埋め込まれるデータを移動又は削除することはできいるが、埋め込まれるデータがテキストフォーマットでない限りそれを変更することはできない。従って、ユーザが、例えば、複合文書１０１内にある予算データ１０３を変更したい場合には、ユーザは、スプレッドシートプログラム２０４をスタータさせ、予算データ１０３をファイルからロードし、変更を行い、その

変更をクリップボード２０３にコピーし、ワードプロセスプログラム２０６をスタートさせ、複合文書１０１をロードし、そして変更されたクリップボードデータを複合文書１０１へペーストする。スプレッドシートプログラムは、スプレッドシートデータを「具現化」し、即ちスプレッドシートプログラムを用いてスプレッドシートフォーマットのデータを操作することができる。プログラムが具現化するフォーマットを「ネーティブ」フォーマットと称する。

公知のシステムは、データを実際に埋め込むのではなく、複合文書に含まれるべきデータに対するリンクを記憶する。ワードプロセスプログラムがデータをクリップボードから複合文書へペーストするときには、複合文書にリンクが記憶される。このリンクは、含まれるべきデータ（通常はファイルに存在する）を指すものである。これらの公知システムは、通常、データに対するリンクを、ワードプロセスプログラムがプレゼンテーションフォーマットとして確認又は処理するフオーマットで与える。例えば、ワードプロセスプログラム２０６が、ユーザにより、スケジュールデータ及び予算データを複合文書へ埋め込みではなくてリンクによってペーストするように指示されるときには、スケジュールデータ及び予算データがプレゼンテーションフォーマットで存在するファイルの名前が文書に挿入される。データの１つのコピーを多数の複合文書により共用できるようにするために、多数の複合文書が同じデータに対するリンクを含むことができる。

発明の要旨

本発明の目的は、コンテナー（収容体）オブジェクトのコンテナーアプリケーションのウインドウ環境内に収容されたオブジェクトと対話するための方法及びシステムを提供することである。

本発明の別の方法は、コンテナーアプリケーションのメニューを、収容されたオブジェクトのサーバアプリケーションのメニューと結合するための方法及びシステムを提供することである。

これら及び他の目的は、本発明の以下の説明から明らかとなるように、コンテナーオブジェクト内に収容されたコンテニー（被収容物）オブジェクトと対話するためのコンピュータ方法及びシステムにより達成される。好ましい実施例において、コンテナーオブジェクトは、そのコンテナーオブジェクトと対話するため

のコンテナーリソースを有するコンテナーウインドウ環境を伴うコンテナーアプリケーションを備えている。コンテニーオブジェクトは、そのコンテニーオブジェクトと対話するためのサーバリソースをもつサーバウインドウ環境を伴うサーバアプリケーションを備えている。本発明の方法は、コンテナーウインドウ環境をディスプレイ装置に表示する。次いで、ユーザは、コンテニーオブジェクトを選択する。コンテニーオブジェクトの選択に応答して、この方法は、複数のサーバリソースを、表示されたコンテナーウインドウ環境で積分する。ユーザがサーバリソースを選択するときには、この方法は、サーバリソース選択を処理するためにサーバアプリケーションを呼び出す。これに対し、ユーザがコンテナーリソースを選択するときには、この方法は、コンテナーリソース選択を処理するためにコンテナーアプリケーションを呼び出す。

図面の簡単な説明

図１は、複合文書の一例を示す図である。

図２は、スケジュールデータ、予算データ及び説明データを複合文書にいかに組み込むかを示す図である。

図３は、図１に示すサンプル複合文書であって、イン・プレース対話を行う前にワードプロセスアプリケーション内で編集されたときに現れる複合文書を示した図である。

図４は、複合文書内の位置でアクチベーションされたときに現れる埋め込まれるスプレッドシートオブジェクトを示す図である。

図５は、オブジェクトハンドラーとコンテナー及びサーバプロセスとの間の関係を示す図である。

図６は、リンクされるか又は埋め込まれるオブジェクトのサンプルインスタンスを示すブロック図である。

図７は、オブジェクトのパブリックビューを示すブロック図である。

図８は、オブジェクトに対して使用できるアクションを表示及び選択するためにコンテナーアプリケーションにより与えられるサンプルユーザメニューを示す図である。

図９は、サーバアプリケーションメニューと図４に示す例のコンテナーアプリ

ケーションメニューとの合体により得られる複合メニューバーを示す図である。

図１０は、本発明の好ましい実施例において複合メニューバーを構成するメニューグループを示す図である。

図１１は、典型的なシングル・ドキュメント・インターフェイスアプリケーションの成分ウインドウを示す図である。

図１２は、マルチプル・ドキュメント・インターフェイスアプリケーションの成分ウインドウを示す図である。

図１３は、埋め込まれたオブジェクトをその位置で編集するときのコンテナーアプリケーションの典型的なウインドウハイアラーキ構成を示すブロック図である。

図１４は、事象駆動のウインドウオペレーティングシステム環境におけるメッセージ処理を示すフローチャートである。

図１４Ｂは、イン・プレース対話をサポートするのに必要なパブリックインターフェイスを示すブロック図である。

図１５は、ＩＯＬＥＩｎＰｌａｃｅＦｒａｍｅ：：ＳｅｔＭｅｎｕ方法の具現化を示すフローチャートである。

図１６は、ＩＯＬＥＩｎＰｌａｃｅＦｒａｍｅ：：ＥｎａｂｌｅＭｏｄｅｌｅｓｓ方法の具現化を示すフローチャートである。

図１７は、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ：：ＯｎＩｎＰｌａｃｅＡｃｔｉｖａｔｅ方法の具現化を示すフローチャートである。

図１８は、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ：：ＯｎＵＩＡｃｔｉｖａｔｅ方法の具現化を示すフローチャートである。

図１９は、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ：：ＯｎＵＩＤｅａｃｔｉｖａｔｅ方法の具現化を示すフローチャートである。

図２０は、ＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：ＩｎＰｌａｃｅＤｅａｃｔｉｖａｔｅ方法の具現化を示すフローチャートである。

図２１は、ＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：ＩｎＰｌａｃｅＵＩＤｅａｃｔｉｖａｔｅ方法の具現化を示すフローチャートである。

図２２は、ＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：Ａｃｔｉｖａｔｅ方法の

具現化を示すフローチャートである。

図２３は、ＡｃｔｉｖａｔｅＵＩファンクションの具現化を示すフローチャートである。

図２４は、ＣｒｅａｔｅＯｂｊｅｃｔＴｏｏｌｂａｒｓファンクションの具現化を示すフローチャートである。

図２５は、図４に示した例に対応する共用メニューデータ構造体のブロック図である。

図２６は、ＯｂｊｅｃｔＳｅｔＭｅｎｕファンクションの具現化を示すフローチャートである。

図２７は、Ｐｒｏｃｅｓｓ＿Ｏｂｊｅｃｔ＿Ａｃｔｉｖａｔｉｏｎファンクションの具現化を示すフローチャートである。

図２８は、ＡＰＩファンクションＯｂｊｅｃｔＬｏａｄをリンクしそして埋め込むオブジェクトの具現化を示すフローチャートである。

図２９は、ＩＯＬＥＯｂｊｅｃｔ：：ＤｏＶｅｒｂ方法の具現化を示すフローチャートである。この方法は、コンテニーオブジェクトと対話する一次方法である。

図３０は、アクチベーション及びデアクチベーションメッセージを処理するためにＭＤＩ文書ウインドウのウインドウ手順によって呼び出されたファンクションＰｒｏｃｅｓｓ＿Ａｃｔｉｖａｔｉｏｎ＿Ｍｅｓｓａｇｅの具現化を示すフローチャートである。

図３１は、Ｐｒｏｃｅｓｓ＿Ｍｏｕｓｅ＿ＬＢｕｔｔｏｎＵｐファンクションの具現化を示すフローチャートである。

好ましい実施例の詳細な説明

目次

１．概要

本発明は、複合文書のコンテクストにおける埋め込まれたデート又はリンクされたデートと対話するための「イン・プレース対話（in-place interaction）」と称する一般的な方法を提供する。即ち、埋め込まれるか又はリンクされたデータと対話するのに使用されるアプリケーションは、複合文書を具現化するアプリケーションのウインドウコンテクスト（メニュー及びウインドウ）を介してユーザにアクセスすることができる。このアクセス性は、「その位置でのアクチベーション（activation in place）」と称する。好ましい実施例において、埋め込まれるか又はリンクされた（収容された）データがその位置でアクチベートされたと

きには、その収容されたデータを具現化するアプリケーションのメニューが、複合文書を具現化するアプリケーションのメニューと合体され、複合メニューバーを形成する。複合メニューバーにおけるメニューの順序は、１組のメニューグループによって決定される。各アプリケーションは、そのメニューをこれらのメニューグループに分類し、そのメニューグループの順序で複合メニューバーにメニューを配置する。次いで、複合メニューバーは、複合文書を具現化するアプリケーションのメニューバーとしてインストールされ、そしてこのアプリケーションのウインドウに送られたフィルタメッセージにメッセージハンドラーがインストールされる。ユーザがメニュー項目を選択するときには、メッセージハンドラーは、そのメニュー項目が、収容されたデータを具現化するアプリケーションのメニューに属するか、又は複合文書を具現化するアプリケーションのメニューに属するかを判断する。次いで、メッセージハンドラーは、その選択されたメッセージ項目に対応する入力メッセージを正しいアプリケーションに送信する。

本発明は、収容されたデータをその位置でアクチベートする１組の抽象クラス（インターフェイス）及びファンクションを定義する。（Ｃ＋＋プログラミング言語において、抽象クラスとは、そのデータ及び方法の定義はもつが、これら方法に対する具現化をもたないクラスである。クラスインスタンスデータを操作するのに使用できる方法に対する実際のコードを与えるのは、クラスを具現化するアプリケーションの役目である。）複合文書を具現化するアプリケーションは、これらインターフェイスの幾つかを具現化する役目を果たし、そして収容されたデータを具現化するアプリケーションは、他のものを具現化する役目を果たす。

本発明の好ましい実施例において、複合文書を形成するアプリケーションプログラムは、別のアプリケーションにより発生されたリンクされるか又は埋め込まれるデータの操作を制御する。オブジェクト指向の用語においては、このデータをオブジェクトと称する。（参照文献Ｂｕｄｄ．Ｔ．　「オブジェクト指向のプログラミングの紹介（An Introduction to Object-Oriented Programming）」アジソン・ウェスリー・パブリッシング社、１９９１年は、オブジェクト指向の概念及び用語を紹介している。）複合文書にリンクされるか又は埋め込まれるオブジェクトは、文書内に「収容」される。又、複合文書は「コンテナー」オブジェ

クト

と称し、そして複合文書内に収容されるオブジェクトは「収容」又は「コンテニー」オブジェクトと称する。図１及び２を参照すれば、スケジュールデータ１０２及び予算データ１０３はコンテニーオブジェクトであり、そして複合文書１０１はコンテナーオブジェクトである。ユーザは、予算データ１０３のようなコンテニーオブジェクトをユーザが編集したい旨をワードプロセッサに指示することができる。予算データ１０３を編集すべきことをユーザが指示すると、ワードプロセスプログラムは、その予算データを編集するのにどのアプリケーションを使用すべきか（例えば、スプレッドシートプログラム）を決定し、そしてそのアプリケーションに着手（スタートアップ）する。次いで、ユーザは、その着手されたアプリケーションを用いて予算データを操作することができ、その変更が複合文書に反映される。予算データが埋め込まれるオブジェクトとして記憶されるかリンクされるオブジェクトとして記憶されるかにかかわりなく、同じ手順が使用される。

予算データを編集するのに用いられるアプリケーションがイン・プレース対話をサポートする場合には、それがワードプロセスプログラムによって着手されたときに、ワードプロセスプログラムのウインドウ環境内でアクチベートされる。図３及び４は、埋め込まれる予算データをその位置でアクチベートするプロセスを示している。

図３は、図１に示すサンプル複合文書であって、イン・プレース対話が行われる前にワードプロセスアプリケーション内で編集されたときに現れる複合文書を示す図である。コンテナーアプリケーション３０１のメインウインドウは、タイトルバー３０２、メニューバー３０３及びクライエントウインドウ３０４を含んでいる。クライエントウインドウ３０４は、図１について述べた製造プロジェクトレポートを表示する。複合文書は、埋め込まれたスプレッドシートオブジェクト（予算データ３０５）を含む。ユーザが複合文書のネーティブテキストデータを編集するときには、メニューバー３０３は、図示したように現れる。即ち、これは、ワードプロセスアプリケーションと対話するに必要な全てのコマンドを含

む。

ユーザは、予算データ３０５を編集することを決定すると、スプレッドシートオブジェクト３０５を選択し、そしてオブジェクトを編集するようにワードプロセスアプリケーションに要求する（例えば、マウスを用いたダブルクリックにより）。ワードプロセスアプリケーションは、次いで、スプレッドシートアプリケーションに着手して、スプレッドシートオブジェクト３０５を編集するよう要求する。スプレッドシートアプリケーションは、ワードプロセスアプリケーションのウインドウ３０１及び３０４とメニューバー３０３とを用いてスプレッドシートオブジェクト３０５を編集するようにワードプロセスアプリケーションと交渉する。

図４は、複合文書内の位置においてアクチベートされたときに現れる埋め込まれたスプレッドシートオブジェクトを示す図である。スプレッドシートオブジェクト４０５は、ワードプロセスアプリケーションのクライエントウインドウ４０４において直接編集される。タイトルバー４０２は、複合文書を具現化するアプリケーション、この場合はワードプロセスアプリケーションが、複合文書「ＶＡＣ１．ＤＯＣ」内のスプレッドシートワークシートを編集することを反映するように変更される。又、メニューバー４０３は、ワードプロセスアプリケーションからのメニューと、スプレッドシートアプリケーションからのメニューとを含む新たな複合メニューバーに変更される。更に、埋め込まれたスプレッドシートオブジェクト４０５の種々の特徴は、それがコンテナー複合文書内で編集されていることを反映するように変更される。陰影付けされた境界パターンの形態である選択ハイライト４０６がオブジェクトの周りに配置される。又、スプレッドシートアプリケーションの標準ツール、この場合は行列マーク４０７がスプレッドシートオブジェクトの周りに配置される。又、現在選択されたセルの周りにはスプレッドシート選択カーソル４０８が配置される。この点において、ユーザは、全てのスプレッドシートアプリケーションコマンドを用いてスプレッドシートオブジェクト４０５を編集する準備ができる。

好ましい実施例において、アプリケーションプログラム（アプリケーション）

は、オブジェクトをリンクし及び埋め込む構成体を用いて互いに共働し、複合文書の形成及び操作を行う。複合文書を形成するアプリケーションは、コンテナー（又はクライエント）アプリケーションと称し、そしてコンテニーオブジェクト

を形成及び操作するアプリケーションは、サーバアプリケーションと称する。アプリケーションは、コンテナー及びサーバの両方として働くことができる。即ちアプリケーションはオブジェクトを含むことができ、そしてアプリケーションが具現化するオブジェクトは他のオブジェクト内に含むことができる。図２を参照すれば、オブジェクトマネージメントプログラム２０１及びスプレッドシートプログラム２０４はサーバアプリケーションであり、そしてワードプロセスプログラム２０６はコンテナーアプリケーションである。コンテナーアプリケーションは、コンテナーオブジェクト内で種々のオブジェクトを選択しそしてコンテニーオブジェクトを操作するための適切なサーバアプリケーションを呼び出すという役目を果たす。サーバアプリケーションは、コンテニーオブジェクトの内容を操作する役目を果たす。

好ましい実施例において、アプリケーションには、具現化とは独立したアプリケーションプログラミングインターフェイス（ＡＰＩ）が与えられ、これは、オブジェクトをリンクしそして埋め込む機能を果たすものである。ＡＰＩは、コンテナー及びサーバアプリケーションによって呼び出される１組のファンクションである。これらのファンクションは、他のものの中でも、コンテナーアプリケーションがサーバアプリケーションとの間でメッセージ及びデータをやり取りするのに必要な設定及び初期化を管理する。

サーバアプリケーションの呼び出しは、サーバアプリケーションがコンテナーアプリケーションとは個別のプロセスとして実行されるときには比較的低速である。ある状態においては、この低速さは特に望ましくない。例えば、ユーザが、多数のコンテニーオブジェクトを含む複合文書をプリントしようとする場合は、各コンテニーオブジェクトに対するサーバプロセスを呼び出してオブジェクトをプリントするよう各サーバプロセスに要求するのに、受け入れられないほど長い時間がかかる。この受け入れられない性能を改善するために、サーバアプリケー

ションは、ランタイム中に、コンテナーアプリケーションのプロセスに動的にリンクして更に効率的なある機能を発揮できるようにするコードを与えることができる。このコードは「オブジェクトハンドラー」と称する。このオブジェクトハンドラーは、オブジェクトをリンクし埋め込むＡＰＩがサーバプロセスをスタートしてサーバプロセスにメッセージを通すのを回避できるようにサーバアプリケーションに代わる機能を発揮する。上記の例では、オブジェクトハンドラーは、オブジェクトをリンクし埋め込むＡＰＩがコンテニーオブジェクトのプリントを呼び出しできるようなプリントファンクションを与えることができる。

図５は、オブジェクトハンドラーとコンテナー及びサーバプロセスとの間の関係を示す図である。オブジェクトハンドラー５０２は、ランタイム中に、オブジェクトリンク及び埋め込みＡＰＩ５０３によってコンテナープロセスアドレススペース５０１にリンクされる。典型的に、オブジェクトリンク及び埋め込みＡＰＩ５０３は、オブジェクトハンドラー５０２を直接呼び出し、そしてコンテナーアプリケーションコードは、ハンドラーがサーバプロセス５０７以外の機能を与えることに気付く必要はない。

１組のファンクションを与えるのに加えて、オブジェクトリンク及び埋め込み（ＯＬＥ）ＡＰＩは、コンテナーアプリケーションがそれらの収容されたオブジェクトと通信できるようにする「インターフェイス」を定義する。インターフェイスは、ある入力、出力及び振る舞いルールにより遵守する１組の方法（Ｃ＋＋言語での）である。収容されたオブジェクトが特定のインターフェイスをサポートする場合には、コンテナーアプリケーションは、そのインターフェイスの方法を呼び出して、定義された振る舞いを行うことができる。特定の実施例では、コンテナーアプリケーションは、オブジェクトデータを直接アクセスしない。むしろ、サポートされたインターフェイスを用いてオブジェクトデータをアクセスするのが好ましい。コンテナーアプリケーションは、インターフェイスに対するポインタを介して収容されたオブジェクトへ結合される。コンテナーアプリケーションは、インターフェイスの方法を呼び出すことによりオブジェクトデータをアクセスする。オブジェクトデータをアクセスするために、これら方法は、指定の

アクセスを要求するサーバアプリケーションへメッセージを送信する。好ましい実施例において、メッセージは、サーバアプリケーションがその基礎となるオペレーティングシステムにより与えられるプロセス間通信メカニズムを用いて個別のプロセスとして具現化されるときにコンテナーアプリケーションとサーバアプリケーションとの間に送られる。

図６は、リンクされるか又は埋め込まれたオブジェクトのサンプルインスタンスを示すブロック図である。好ましい実施例において、インスタンスのレイアウトは、参考としてここに取り上げる「オブジェクト指向のプログラミング言語のためのコンパイラーにおいて仮想ファンクション及び仮想ベースを具現化するための方法（A Method for Implementing Virtual Functions and Virtual Bases in a Compiler for an Object Oriented Programming Lang）」と題する米国特許出願第０７／６８２，５３７号に規定されたモデルに合致する。このインスタンスは、各々のサポートされるインターフェイスに対するオブジェクトデータ構造６０１及びインターフェイスデータ構造６１を含んでいる。オブジェクトデータ構造６０１は、インターフェイスデータ構造６１３に対するポインタ６０２を含み、そしてインスタンスのプライベートデータを含んでもよい。このサンプルインスタンスのプライベートデータは、クラス識別子６０３と、オブジェクトの記憶に対するハンドル６０４と、オブジェクトの状態を追跡するためのデータ６０５とを備えている。クラス識別子（ＣＬＡＳＳ＿ＩＤ）は、オブジェクトに対する適切なサーバアプリケーションをアクセスするのに使用される。これは、プログラミング言語に使用されるデータ構造「タイプ」と同様である。インターフェイスは、ＣＬＡＳＳ＿ＩＤを永続的なグローバルな登録簿へのインデックスとして用いることによりオブジェクトに対するサーバアプリケーションを決定することができる。永続的なグローバルな登録簿は以下で詳細に説明する。図６に示すように、各インターフェイスデータ構造６１３は、プライベートデータ構造６０６及び仮想ファンクションテーブル６０８を含む。プライベートデータ構造６０６は、仮想ファンクションテーブル６０８に対するポインタ６０７を含む。仮想ファンクションテーブル６０８は、インターフェイスの方法を具現化するコー

ド６１０、６１２に対するポインタ６０９、６１１を含む。

テーブル１

```
# define interface class
interface intf   {public:
    virtual RETCODE fnc1(arg1, arg2)=0;
    virtual RETCODE fnc2(arg1, arg2)=0;
      virtual RETCODE fnc3()=0;
                ...
      } ;
```

テーブル１は、オブジェクトデータ構造６０１における第１エントリーｐｉｎｔｆ，に対するインターフェイスの定義を表している。テーブル１において、語「インターフェイス」は、Ｃ＋＋クラスを意味するものと定義される。この定義は、３つの方法をそれらのパラメータと共に示している。各パラメータリストの端にある「＝０」は、その方法がコードの具現化をもたないことを意味する。Ｃ＋＋プログラミング言語において、これらのファンクションは「純粋な仮想ファンクション」と称される。純粋な仮想ファンクションをもつクラスは、抽象クラスと称される。

図７は、オブジェクトのパブリックビューを示すブロック図である。オブジェクトのパブリックビューは、オブジェクトが７０２ないし７０６をサポートする種々のインターフェイスである。各インターフェイスは、コンテナアプリケーションがオブジェクトをアクセスできるようにする方法を与える。各オブジェクトは、ＩＵｎｋｎｏｗｎインターフェイス７０２をサポートする。コンテナーアプリケーションは、ＩＵｎｋｎｏｗｎインターフェイス７０２を使用して、オブジェクトが他のインターフェイスのどれをサポートするかを決定する。特定のオブジェクトに対するＩＵｎｋｎｏｗｎインターフェイス７０２の具現化は、オブジェクトが他のどんなインターフェイスをサポートするかを知り、そしてそれらインターフェイスに対する呼び出しアプリケーションポインタへ復帰する。好ましい実施例では、方法ＩＵｎｋｎｏｗｎ：：ＱｕｅｒｙＩｎｔｅｒｆａｃｅがこの

目的に使用される。インターフェイス７０３ないし７０６は、オブジェクトによってサポートすることのできる典型的なインターフェイスの例である。これらのインターフェイスは、ＩＵｎｋｎｏｗｎインターフェイスから導出される。例えば、ＩＤａｔａＯｂｊｅｃｔインターフェイス７０３は、データを記憶すると共にオブジェクトからデータを検索するための方法を与える。ＩＯＬＥＣｏｎｔａｉｎｅｒ７０４は、オブジェクト内に収容されるコンテニーオブジェクトをリストするための方法を与える。ＩＰｅｒｓｉｓｔＳｔｏｒａｇｅインターフェイス７０５は、オブジェクトを永続的な記憶装置に記憶したりそこからオブジェクトを検索したりする方法を与える。ＩＯＬＥＯｂｊｅｃｔインターフェイス７０６は、コンテナーアプリケーションがユーザ選択アクションに対応するオブジェクトの機能を呼び出すところの方法を与える。

ＡＰＩに加えて、本発明のオブジェクトリンク及び埋め込み構成体は、永続的なグローバルな「登録簿」によりコンテナー及びサーバアプリケーションへ情報を与える。この登録簿は、（１）オブジェクトの各タイプごとに、オブジェクトタイプを具現化するサーバアプリケーション、（２）各サーバアプリケーションがコンテナーアプリケーションに与えるアクション、（３）各サーバアプリケーションに対する実行可能なファイルが置かれた場所、及び（４）各サーバアプリケーションがそれに関連したオブジェクトハンドラーを有するかどうか、といった情報のデータベースである。

２．<u>イン・プレース対話の概要</u>

オブジェクトがいったん文書にリンクされるか又は埋め込まれると、ユーザはオブジェクトを選択しそしてその選択されたオブジェクトに基づいてあるアクションを実行することを要求する。ユーザは、先ず、オブジェクトを選択しそしてオブジェクトに基づいて実行されるべきアクション（例えば、メニュー項目）を選択することによりアクションを要求する。次いで、具現化するサーバアプリケーションが呼び出され、その選択されたアクションを実行する。考えられるアクションの選択をユーザに表示しそしてユーザがアクションを選択できるようにする方法は多数あることが当業者に明らかであろう。好ましい実施例では、コンテ

ナーアプリケーションは、グローバルな登録簿から、その選択されたオブジェクトを具現化するサーバアプリケーションによってどんなアクションがサポートされるかを判断し、そしてアクションをメニューに表示する。

図８は、オブジェクトに対して使用できるアクションを表示しそして選択するためにコンテナーアプリケーションによって与えられる例示的なユーザメニューを示す。メニュー項目８０３は、コンテナーアプリケーションＥｄｉｔ（編集）メニュー８０２におけるオブジェクトのエントリーである。このエントリーは、現在選択されたオブジェクトに基づいて変化する。埋め込まれるか又はリンクされたオブジェクトが選択されないときは、メニュー項目８０３は表示されない。サブメニュー８０４は、「エクセル・ワークシート・オブジェクト（Excel Worksheet Object）」によってサポートされるアクションを表示する。この例では、サポートされるアクションは、「ＥｄｉＵＮ　「Ｏｐｅｎ（オープン）」及び「Ｔｙｐｅ（タイプ）」である。サブメニューの最初のアクション（例えば、「Ｅｄｉｔ」）は、ユーザがオブジェクトのマウスポインタ装置でダブルクリックするか又は機能的に等価なキーを入力するときに実行されるデフォールトアクションである。

ユーザが所望のアクションを選択すると（メニューから又はオブジェクトにおけるダブルクリック動作により）、コンテナーアプリケーションは、サーバアプリケーションを呼び出し、コンテナーアプリケーションに代わってどんなアクションを実行すべきかをそこに通す。コンテナーアプリケーションは、オブジェクトに対するＩＯＬＥＯｂｊｅｃｔインターフェイスを得そしてオブジェクトのＤｏＶｅｒｂ方法を呼び出して、選択されたアクションをそこに通すことにより、これを行う。（ＤｏＶｅｒｂ方法は、オブジェクトにおいてオブジェクトに特定のアクションを実行する。）サーバアプリケーションは、次いで、コンテナアプリケーションのコンテクスト内の位置でオブジェクトをアクチベートできるかどうかを決定する。もしそうならば、サーバアプリケーション及びコンテナアプリケーションは、それらのメニューを複合メニューバーに合体し、サーバアプリケーションツールバー、パレット、公式等の配置を交渉し、そして合体されたメッ

セージの取り扱いを設定する。この点において、サーバアプリケーションは、ユーザ入力を受け取る準備ができる。

図４の例を続けると、図示されたように、ユーザは、ワードプロセスアプリケーションのウインドウ環境内の位置においてスプレッドシートオブジェクト（予算データ４０５）を編集する。図９は、サーバアプリケーションのメニューと、図４に示す例のコンテナーアプリケーションメニューとの合体により生じる複合メニューバーを示す図である。複合メニューバー９０１は、ワードプロセスアプリケーションからのメニュー９０２、９０５と、スプレッドシートアプリケーションからのメニュー９０３、９０４、９０６とを備えている。ユーザがこれらメニューの１つから特定のメニュー項目を選択すると、コンテナーアプリケーションは、合体されたメッセージハンドラーにより、メッセージをワードプロセスアプリケーションへ発送すべきか、スプレッドシートアプリケーションへ発送すべきかを決定する。

本発明の好ましい実施例では、複合メニューバーは、１組の所定の規定に基づいて形成される。複合メニューバーに含まれるべき各アプリケーションメニューは、メニューグループに指定される。次いで、メニューは、その指定されたメニューグループに基づいて複合メニューバーに挿入される。

図１０は、本発明の好ましい実施例において複合メニューバーを構成するメニューグループを示す図である。複合メニューバー１００３は、コンテナーアプリケーションからのメニューグループ１００１と、サーバアプリケーションからのメニューグループ１００２とを備えている。コンテナアプリケーションのメニューグループ１００１は、ファイルグループと、コンテナーグループと、ウインドウグループとを含む。サーバアプリケーションのメニューグループ１００２は、編集グループと、オブジェクトグループと、ヘルプグループとを含む。好ましい実施例において、コンテー及びサーバアプリケーションのメニューは、参考としてここに取り上げる１９９２年マイクロソフト社の「ザ・ウインドウズ・インターフエイス：アプリケーションデザインガイド（The Windows Interface: An Application Design Guide）」に規定されたマイクロソフトアプリケーションユー

ザインターフェイススタイルガイドラインに基づいて最終的な複合メニューバーにインターリーブされる。特に、複合メニューバー１００３において、グループは、次の順序、即ちファイル、編集、コンテナー、オブジェクト、ウインドウ及びヘルプの順に、左から右へ配列される。

３．イン・プレース対話のウインドウサポート

好ましい実施例においては、イン・プレース対話ＡＰＩは、その基礎となるウインドウシステムの能力を用いて具現化される。本発明は、その基礎となるウインドウシステムがマイクロソフトウインドウズ３．１オペレーティングシステム（"Windows"）に類似したものであると仮定して説明するが、本発明は、異なる基礎的なウインドウシステムでも実施できることが当業者に明らかであろう。マイ

クロソフトウインドウズ３．１オペレーティングシステムは、参考としてここに取り上げる１９９２年マイクロソフト社の「プログラマの参考書、第２巻：ファンクション（Programmer's Reference,Volume 2: Functions）」；１９９２年マイクロソフト社の「プログラマの参考書、第３巻：メッセージ、構造及びマクロ（Programmer's Reference,Volume 3: Messages,Structures,and Macros）」：及び１９９２年マイクロソフト社の「プログラミングの手引き（Guide to Programming）」に掲載されている。

ウインドウ環境では、アプリケーションは、単一文書インターフェイス又は多文書インターフェイスをサポートする。単一文書インターフェイス（ＳＤＩ）アプリケーションは、一度に１つの文書（ファイル）と対話する。例えば、ＳＤＩをサポートするワードプロセスアプリケーションは、現在編集されているファイルをその一次ウインドウに表示する。多文書インターフェイス（ＭＤＩ）のアプリケーションは、各文書に対して少なくとも１つのウインドウを割り当てることにより多数の文書（ファイル）と対話する。例えば、ＭＤＩをサポートするワードプロセスアプリケーションは、現在編集されている各ファイルを個別の文書ウインドウに表示する。ユーザは、所望の文書ウインドウのタイトルバーにおいてクリックするか、又はアプリケーションのウインドウメニュー上のリストからウ

インドウタイトルを選択することにより、ユーザが編集しようとするファイルの文書ウインドウを選択する。

図１１は、典型的な単一文書インターフェイスアプリケーションの成分ウインドウを示す図である。典型的なＳＤＩアプリケーションは、フレームウインドウ１１０１を与え、そしてアプリケーションによっては、埋め込まれたオブジェクトの存在に対し区画ウインドウ１１０５及び１１０６と、親ウインドウ１１０７とを更に与えることができる。ＳＤＩアプリケーションの場合には、フレームウインドウ１１０１は文書ウインドウでもある。区画ウインドウ１１０５、１１０６は、複合文書の多数のビューを与える。親ウインドウ１１０７は、オブジェクトが複合文書に最初に挿入されたときにオブジェクトの輪郭を描くようにコンテナーアプリケーションによって形成される。図１１に示す例では、埋め込まれたオブジェクトはスプレッドシートオブジェクトであり、これは、コンテナーアプリケーションの親ウインドウ１１０７内に収容されたオブジェクトウインドウ１１０８内に表示される。オブジェクトウインドウ１１０８は、サーバアプリケーションによって所有される。フレームウインドウ１１０１は、タイトルバー１１０２と、メニューバー１１０３と、ツールバー１１０４とを含む。典型的に、ツールバー及びその他のアプリケーションに特定のツールは、フレームウインドウに取り付けられるか、又はコンテナーアプリケーションの区画ウインドウに取り付けられる。又、これらは、図１１に示すウインドウとは独立したウインドウであるフローティングパレットとしても現れ、従って、頂部に「浮動」するように見える。

図１２は、典型的な多文書インターフェイスアプリケーションの成分ウインドウを示す図である。典型的なＭＤＩアプリケーションは、ユーザが同じコンテナーアプリケーション内から多数の成分文書を編集できるようにする。図１２に示す例では、ユーザは、２つの文書ウインドウ１２０５、１２０６において２つの個別の成分文書を編集する。各文書ウインドウは、ＳＤＩアプリケーションと同様に、区画ウインドウを含むことができる。文書ウインドウ１２０５は、２つの区画ウインドウ１２０７、１２０８を含む。又、ＭＤＩアプリケーションは、Ｓ

ＤＩアプリケーションと同様に、埋め込まれたオブジェクトを収容するための親ウインドウ１２０９を与えることができる。図１２は、オブジェクトウインドウ１２１０内に存在する埋め込まれたスプレッドシートオブジェクトを示す。ＳＤＩアプリケーションの場合と同様に、アプリケーションに特定のツールがどこかに現れる。

ＳＤＩ又はMDＩのいずれかのアプリケーションにより管理されるウインドウは、ハイアラーキ形態で形成されそして維持される。図１３は、埋め込まれたオブジェクトをその位置において編集するときのコンテナーアプリケーションの典型的なウインドウハイアラーキを示すブロック図である。このウインドウハイアラーキは、コンテナーアプリケーションからのコンテナーアプリケーションウインドウ１３０１と、サーバアプリケーションからのサーバアプリケーションウインドウ１３０７とを備えている。コンテナーアプリケーション１３０２は、そのフレームウインドウ１３０３を管理し、該ウインドウは文書ウインドウ１３０４

を含み、該ウインドウは区画ウインドウ１３０５を含み、そして該ウインドウは親ウインドウ１３０６を含む。オブジェクトがその位置においてアクチベートされるときは、サーバアプリケーション１３０８は、埋め込まれたオブジェクトに対するルートウインドウ１３０９と、これが必要とする子ウインドウとを形成する。オブジェクトルートウインドウ１３０９は、オブジェクト子ウインドウ１３１０、１３１１、１３１２を含む。

各アプリケーションは、個別のプロセスとして具現化されるときに、アプリケーションのウインドウハイアラーキに存在するウインドウに接続された事象を受け取るための入力待ち行列を含んでいる。図１３のウインドウハイアラーキは、２つの異なるアプリケーションによってサポートされる。従って、コンテナーアプリケーションに属するウインドウと、サーバアプリケーションに属するウインドウとに関連した個別の入力待ち行列がある。入力待ち行列１３１３は、コンテナーアプリケーションウインドウ１３０１に関連している。入力待ち行列１３１４は、サーバアプリケーションウインドウ１３０７に関連している。ユーザがマウスでクリックするか、又はこれらウインドウの１つに対してキーストロークを

入力したときに、その基礎となるウインドウシステムは、コンテナー入力待ち行列１３１３又はサーバアプリケーション待ち行列１３１４のいずれかに適当なメッセージを入力する。

４．イン・プレース対話ＡＰＩ

オブジェクトリンク及び埋め込みＡＰＩは、イン・プレース対話をサポートするためにコンテナーアプリケーションとサーバアプリケーションが通信するところのファンクションを与え、そのインターフェイスを定義する。これらインターフェイスの方法及びその他のＡＰＩファンクションは、ユーザ入力の通常の処理中にアプリケーションコードによって呼び出される。事象駆動されるウインドウシステムにおいては、アプリケーションは、ユーザが特定のメニュー項目又はオブジェクトを選択したことを指示するメッセージを受け取るのに応答して適当な方法又はファンクションを呼び出す。

図１４は、事象駆動式ウインドウオペレーティングシステム環境におけるメッセージの処理を示すフローチャートである。各ウインドウは、そのウインドウが

形成されるときにその基礎となるウインドウシステムと整列されるそれ自身のメッセージハンドラーを有している。アプリケーション入力待ち行列（例えば、コンテナーアプリケーションの入力待ち行列１３１３）にメッセージが受け取られると、アプリケーションは、メッセージをフィルタし、変換し、又はウインドウシステムへ発送する。ウインドウシステムのディスパッチャーは、次いで、メッセージにおいて指示された特定のウインドウに対して既に整列されているメッセージ取扱ファンクション（メッセージハンドラー）にメッセージを送信する。メッセージを受け取ると、メッセージハンドラーは、メッセージを処理する。この処理は、オブジェクトリンク及び埋め込みＡＰＩの使用を含む。ステップ１４０１及び１４０２は、メッセージポンプを形成する。ステップ１４０１において、アプリケーションは、その入力待ち行列におけるメッセージを待機する。ステップ１４０２において、アプリケーションは、メッセージを適宜フィルタ又は変換するか、又はメッセージをウインドウシステムのディスパッチファンクションへ発送する。ステップ１４０３及び１４０４は、ウインドウシステムのディスパッ

チファンクションにおいて、メッセージを適当なウインドウメッセージハンドラーへ発送するステップである。ステップ１４０３において、ウインドウシステムディスパッチャーは、メッセージにおいて指示されたウインドウに対しメッセージハンドラーを位置決めする。ステップ１４０４において、ウインドウシステムディスパッチャーは、その位置決めされたメッセージハンドラーへメッセージを送信する（例えば、メッセージハンドラーを呼び出すことにより）。

ステップ１４０５ないし１４１２は、ウインドウに対する典型的なメッセージハンドラーを構成する。メッセージハンドラーは、「ウインドウ手順」とも称する。好ましい実施例において、アプリケーションが特定のウインドウに対するウインドウ手順を与えない場合には、その基礎となるウインドウシステムは、ＤｅｆＷｉｎｄｏｗＰｒｏｃと称するデフォールトウインドウハンドラーを与える。ステップ１４０５ないし１４０８において、アプリケーションは、メッセージをデコードして、どのタイプの事象が生じたかを決定する。典型的に、各タイプの事象ごとに、アプリケーションは、ステップ１４０９ないし１４１２に示すように異なるファンクションを呼び出す。これらのファンクションは、次いで、オブジェクトリンク及び埋め込みＡＰＩを使用する。例えば、メニュー事象が受け取られたときには、アプリケーションは、ステップ１４１１において、メニュー事象を処理するファンクションを呼び出す。ステップ１４１１は、コンテニーオブジェクトをその位置においてアクチベートするＰｒｏｃｅｓｓ＿ＯｂｊｅｃｔＡｃｔｉｖａｔｉｏｎファンクション（ステップ１４１３として示す）を呼び出す。以下に詳細に述べるように、Ｐｒｏｃｅｓｓ＿Ｏｂｊｅｃｔ＿Ａｃｔｉｖａｔｉｏｎファンクションは、オブジェクトリンク及び埋め込みＡＰＩを用いて、コンテニーオブジェクトをアクチベートする。

イン・プレース対話ＡＰＩは、次のインターフェイス、即ちＩＯＬＥＷｉｎｄｏｗ；ＩＯＬＥＩｎＰｌａｃｅＵＩＷｉｎｄｏｗ；ＩＯＬＥＩｎＰｌａｃｅＦｒａｍｅ；ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ；及びＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔを定義する。ＩＯＬＥＷｉｎｄｏｗインターフェイスは、他のインターフェイスの１つに関連したウインドウハンドルを検索するための方法を与える

ものである。ＩＯＬＥＩｎＰｌａｃｅＵＩＷｉｎｄｏｗインターフェイスは、サーバアプリケーションがウインドウツールの配置に対してコンテナーアプリケーションと交渉する方法を与える。ＩＯＬＥＩｎＰｌａｃｅＦｒａｍｅインターフェイスは、サーバアプリケーションがコンテナーアプリケーションのフレームウインドウと通信する方法を与える。ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔインターフェイスは、サーバアプリケーションがコンテナーアプリケーションの親ウインドウと通信する方法を与える。ＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイスは、コンテナーアプリケーションがサーバアプリケーションをアクチベート及びデアクチベートする方法を与える。図１４Ｂは、イン・プレース対話をサポートするのに必要なパブリックインターフェイスを示すブロック図である。コンテナーオブジェクト１４Ｂ０１は、ＩＯＬＥＷｉｎｄｏｗインターフェイス１４Ｂ０２、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔインターフェイス１４Ｂ０３及びＩＯＬＥＩｎＰｌａｃｅＦｒａｍｅインターフェイス１４Ｂ０４をサポートする。コンテニーオブジェクト１４Ｂ０５は、ＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイス１４Ｂ０６をサポートする。これらインターフェイスの各々について、以下に詳細に説明する。

４．１　ＩＯＬＥＷｉｎｄｏｗインターフェイス

テーブル２は、ＩＯＬＥＷｉｎｄｏｗインターフェイスをリストしたものである。オブジェクト指向の用語において、ＩＯＬＥＷｉｎｄｏｗインターフェイスは、他のイン・プレース対話インターフェイスの「基本的クラス」である。従って、他のインターフェイスは、ＩＯＬＥＷｉｎｄｏｗインターフェイスから導出され、そのパブリック方法を継承する。ＩＯＬＥＷｉｎｄｏｗインターフェイスには、ＧｅｔＷｉｎｄｏｗと称する１つのパブリック方法しかない。

テーブル２

```
interface IOLEWindow: public IUnknown  {public:
    virtual SCODE GetWindow (HWND FAR*phwnd)=0;
}
```

４．１．１　IOLEWindow::GetWindow

ＧｅｔＷｉｎｄｏｗ方法は、これが呼び出されたＩＯＬＥＩｎＰｌａｃｅＵＩＷｉｎｄｏｗＮＩＯＬＥＩｎＰｌａｃｅＦｒａｍｅ、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ又はＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイスに対応するウインドウハンドル（独特のウインドウ識別子）を検索する。検索されたウインドウハンドルは、通常、その基礎となるウインドウシステムファンクションを呼び出すときに使用される。

４．２　ＩＯＬＥＩｎＰｌａｃｅＵＩＷｉｎｄｏｗインターフェイス

テーブル３は、ＩＯＬＥＩｎＰｌａｃｅＵＩＷｉｎｄｏｗインターフェイスをリストする。ＩＯＬＥＩｎＰｌａｃｅＵＩＷｉｎｄｏｗインターフェイスの方法は、コンテナーアプリケーションの文書及び区画ウインドウ内のツール配置を交渉するためにサーバアプリケーションによって呼び出される。

テーブル３

```
interface IOLEInPlaceUIWindow: public IOLEWindow {public:
    virtual SCODE GetBorder(RECT borderRect)=0;
    virtual SCODE QueryBorderSpace(RECT widthRect)=0;
    virtual SCODE SetBorderSpace(RECT widthRect)=0;
}
```

４．２．１　IOLEInPlaceUIWindow::GetBorder

ＧｅｔＢｏｒｄｅｒ方法は、サーバアプリケーションがそのツール（子ウインドウとして実現される）を配置できるようにする位置を検索する。この方法は、文書インターフェイスが呼び出されるか区画インターフェイスが呼び出されるかに基づいて、文書又は区画ウインドウのフレーム内に配置された四角形を返送する。サーバアプリケーションは、この四角形を検索すると、ツールを配置するのに必要なスペースの巾を決定することができると共に、ＱｕｅｒｙＢｏｒｄｅｒＳｐａｃｅ方法を用いてこのスペースを要求することができる。コンテナーアプリケーションによって返送された四角形が拒絶される場合には、サーバアプリケーションは、その位置でのアクチベーションを続けないように選択することもできるし、或いはそのツールをアクチベートしないように選択することもできる。

４　２．２　IOLEInPlaceUIWindow::QueryBorderSpace

ＱｕｅｒｙＢｏｒｄｅｒＳｐａｃｅ方法は、サーバアプリケーションがそのツールを配置することのできる区画又は文書ウインドウ内の指定量のスペースを検索する。この方法は、サーバアプリケーションがそのツール配置に必要なＧｅｔＢｏｒｄｅｒへの以前の通話から検索された四角形内の１つのパラメータ、即ち四角形のボーダースペース、を取り出す。この方法は、文書又は区画ウインドウが要求を受け入れられるかどうかの指示を返送する。要求を受け入れられない場合には、サーバアプリケーションは、異なる四角形でこの方法を再び呼び出し、その位置でのアクチベーションを続けないように選択し、又はそのツールをアクチベートしないように選択することができる。

４．２．３　IOLEInPlaceUIWindow::SetBorderSpace

ＳｅｔＢｏｒｄｅｒＳｐａｃｅ方法は、それに関連したコンテナアプリケーションに、サーバアプリケーションのツールを配置するために区画ウインドウ又は文書ウインドウに指定のスペースを実際に割り当てていることを知らせる。この方法は、ＱｕｅｒｙＢｏｒｄｅｒＳｐａｃｅへの以前の通話において区画又は文書ウインドウからスペースが首尾良く要求された後に呼び出される。サーバアプリケーションは、それが必要とするスペースを割り当てる役目を果たす。この方法は、サーバアプリケーションがその子ウインドウに割り当てようとする１つのパラメータ、即ち四角形のスペース、を取り出す。指定された四角形は、以前に首尾良く要求されたものより小さくてもよい。「指定された」という用語は、パスしたパラメータを指し、そして「特定された」という用語は、特定の方法が属するインターフェイス、クラス、ウインドウ又はオブジェクトを指す。この方法は、特定された区画又は文書ウインドウのユーザインターフェイスリソースのいずれかを必要に応じて移動し又はサイズ決めする。

４．３　ＩＯＬＥＩｎＰｌａｃｅＦｒａｍｅインターフェイス

テーブル４は、ＩＯＬＥＩｎＰｌａｃｅＦｒａｍｅインターフェイスをリストする。このＩＯＬＥＩｎＰｌａｃｅＦｒａｍｅインターフェイスは、そのコンテナーアプリケーションのフレームウインドウと通信するためにサーバアプリケーションにより呼び出される方法を与える。

テーブル４

```
interface IOLEInPlaceFrame: public IOLEInPlaceUIWindow {public:
    virtual SCODE SetMenu(HANDLE hSharedMenu. HWND hwndObject)=0;
    virtual SCODE InsertMenu(HANDLE hmenu. UNIT FAR*lpiMenuCounts)=0;
    virtual SCODE RemoveMenu(HANDLE hmenu)=0;
    virtual SCODE SetStatusText(LPSTR lpszStatusText)=0;
    virtual SCODE EnableModeless(BOOL fEnable)=0;
    virtual SCODE TranslateAccelerator(LPMSG lpmsg, WORD WID)=0;
}
```

４．３．１ IOEInPlaceFrame::SetMenu

ＳｅｔＭｅｎｕ方法は、指定された複合メニューバーをコンテナーアプリケーションのメニューバーとしてインストールしたり除去したりし、そして複合メニューバーに対するメッセージハンドラーをインストールする。図１５は、IOEInPlceFrame::SetMenu方法の具現化を示すフローチャートである。この方法は、コンテナーアプリケーションがＭＤＩアプリケーションであるかＳＤＩアプリケーションであるかに基づいて異なるメカニズムを使用して複合メニューバーをインストールする。ステップ１５０１において、この方法は、指定された複合メニューバーがＮＵＬＬであるかどうかを決定し、もしそうであれば、ステップ１５０２へ続き、さもなくば、ステップ１５０３へ続く。ステップ１５０２では、この方法は、ヘルパーファンクションＯｂｊｅｃｔＳｅｔＭｅｎｕＤｅｓｃｒｉｐｔｏｒを呼び出して、複合メニューバーに対するメッセージハンドラーを除去しそして復帰する。ステップ１５０３では、この方法は、コンテナーアプリケーションがＳＤＩアプリケーションであるかどうかを決定し、もしそうであれば、ステップ１５０４へ続き、さもなくば、ステップ１５０５へ続く。ステップ１５０４においては、この方法は、その基礎となるウインドウシステムファンクションＳｅｔＭｅｎｕを呼び出し、複合メニューバーをコンテナーアプリケーションフレームウインドウのメニューバーとしてインストールし、そしてステップ１５０７へ続く。ステップ１５０５では、この方法は、フレームウインドウにメッセー

ジを送り、そのＭＤＩメニュー設定を実行するように通知する。ステップ１５０６では、この方法は、その基礎となるウインドウシステムファンクションＤｒａｗＭｅｎｕＢａｒを呼び出してメニューバーを再描写する。ステップ１５０７では、この方法は、ヘルパーファンクションＯｂｊｅｃｔＳｅｔＭｅｎｕＤｅｓｃｒｉｐｔｏｒを呼び出して、複合メニューバーに対するメッセージハンドラーをインストールする。ステップ１５０８では、この方法は、そのメニューバーを切り換えるときに必要となる他のプロセスを実行し、次いで、復帰となる。

４．３．２　IOLEInPlaceFrame::InsertMenus

Code Table 1

```
VOID IOleInPlaceFrame::InsertMenus (hmenu, ContrCounts) {
1   if there are File Group menus present {
2       for each filegroupmenu {
3           hfilemenu = CreateMenu ( );
4           InsertMenu (hmenu, MF_APPEND, MF_BYPOSITION | MF_POPUP, hfilemenu);
5           ContrCounts[0] = ContrCounts[0] + 1;
6           for each filegroupmenu_item {
7               InsertMenu (hfilemenu, MF_APPEND, MF_BYPOSITION | MF_STRING,
8                       item_id, "string to be displayed");
9           }
10      };
11  if there are Container Group menus present {
12      for each containergroupmenu {
13          hcontmenu = CreateMenu ( );
14          InsertMenu (hmenu, MF_APPEND, MF_BYPOSITION | MF_POPUP, hcontmenu);
15          ContrCounts[1] = ContrCounts[1] + 1;
16          for each contgroupmenu_item {
17              InsertMenu (hcontmenu, MF_APPEND, MF_BYPOSITION | MF_STRING,
18                      item_id, "string to be displayed");
19          }
20      };
21  if there are Window Group menus present {
22      for each windowgroupmenu {
23          hwndmenu = CreateMenu ( );
24          InsertMenu (hmenu, MF_APPEND, MF_BYPOSITION | MF_POPUP, hwndmenu);
25          ContrCounts[2] = ContrCounts[2] + 1;
26          for each wndgroupmenu_item {
27              InsertMenu (hwndmenu, MF_APPEND, MF_BYPOSITION | MF_STRING,
28                      item_id, "string to be displayed");
29          }
30      };
31  return ( );
}
```

ＩｎｓｅｒｔＭｅｎｕｓ方法は、コンテナアプリケーションのメニューを、サーバアプリケーションによって形成される複合メニューバーに挿入ずる。コード

テーブル１は、IOLEInPlaceFrame::InsertMenus方法を具現化するための擬似コードを示す。この方法は、複合メニューバーと、メニューカウントのアレーとの２つのパラメータをとる。コンテナアプリケーションのメニューによって表されたメニューグループの各々に対し、そのグループに対するメニューを挿入するループがある。ライン１ないし１０において、何らかのファイルグループメニューがある場合には、この方法は、これらのメニューを複合メニューバーに挿入し、そしてファイルグループに対応するインデックスにおいてメニューカウントアレーを増加する。（例えば、図４の例に示されたメニューバーが使用される場合には、インデックス＝０である。）ライン１１ないし２０において、何らかのコンテナーグループメニューが存在する場合には、この方法は、これらのメニューを複合メニューバーに挿入し、そしてコンテナーグループに対応するインデックスにおいてメニューカウントアレーを増加する。最後に、ライン２１ないし３０において、追加されるべきウインドウグループメニューがある場合には、この方法は、これらメニューを複合メニューバーに挿入し、そしてウインドウグループに対応するインデックスにおいてメニューカウントアレーを増加する。この方法の完了時に、各インデックスにおいてメニューカウントアレーに記憶される値は、その特定のメニューグループに対してコンテナーアプリケーションが挿入したメニューの数を指示する。この方法は、その基礎となるウインドウシステムから標準的なファンクション（ＣｒｅａｔｅＭｅｎｕ及びＩｎｓｅｒｔＭｅｎｕ）を呼び出し、コンテナーアプリケーションに対するメニューを形成すると共に、それらを複合メニューバーに挿入する。

４．３．３　IOLEInPlaceFrame::RemoveMenus

ＲｅｍｏｖｅＭｅｎｕｓ方法は、サーバアプリケーションが複合メニューバーの割り当てを解除する前に、コンテナーアプリケーションがそのメニューを複合メニューバーから除去できるようにする。この方法は、IOLEInPaceObject::InPlaceDeactive方法から呼び出される。ＲｅｍｏｖｅＭｅｎｕｓ方法は、１つのパラメータ、即ちコンテナーメニューが記憶される複合メニューバーのハンドルを取り出す。複合メニューバーは、この方法が呼び出される前に全てのサーバメニ

ューをクリアすることが予想される。

４．３．４　IOLEInPlaceFrame::SetStatusText

ＳｅｔＳｔａｔｕｓＴｅｘｔ方法は、サーバアプリケーションがコンテナーアプリケーションのフレームウインドウの状態ウインドウ（もし１つあれば）をセットできるようにする。通常は、状態ウインドウは、フレームウインドウの底部に配置され、現在選択に対応する状態又はヒント情報を含んでいる。ＳｅｔＳｔａｔｕｓＴｅｘｔ方法は、コンテナーアプリケーションに特定のもので、コンテナーアプリケーションがその状態ウインドウをセットするために通常行うどんなオペレーションも実行する。この方法は、１つのパラメータ、即ち状態ウインドウに挿入するためのテキストストリングを取り出す。

４．３．５　IOLEInPlaceFrame::EnableModeless

ＥｎａｂｌｅＭｏｄｅｌｅｓｓ方法は、コンテナーアプリケーションに対して現在表示されているモードレスダイアログをイネーブル又はディスエイブルするものである。モードレスダイアログとは、ユーザによって明確に閉じられるまで表示される入力ウインドウである。このウインドウが表示される間に、ユーザは他のウインドウと対話することができる。一方、モードダイアログは、ユーザが受け入れられる入力を入れるまで他のウインドウ処理を阻止する入力ウインドウである。この方法は、モードダイアログを表示したいが、それに関連したコンテナーアプリケーションがモードレスダイアログを既に表示しているときにサーバアプリケーションによって呼び出される。

図１６は、IOLEInPlaceFrame::EnableModeless方法を具現化するフローチャートである。この方法は、コンテナーアプリケーションのモードレスダイアログを隠し、そして再び呼び出されたときに、モードレスダイアログを復帰させる。指定されたフラグｆＥｎａｂｌｅが真である場合には、その隠されたダイアログが表示され、さもなくば、現在表示されているいかなるモードレスダイアログも隠される（表示から除去されるが、内部メモリのデータ構造は割り当て解除されない）。ステップ１６０１において、この方法は、ＦＥｎａｂｌｅが真であるかどうか決定し、もしそうならば、ステップ１６０２へ続くが、さもなくば、ステッ

プ１６０３へ続く。ステップ１６０２において、この方法は、その基礎となるウインドウシステムファンクションＳｈｏｗＷｉｎｄｏｗを呼び出し、既にセーブされたモードレスダイアログに関連したウインドウを回復させ、次いで、復帰する。ステップ１６０３において、この方法は、現在表示されている次のモードレスダイアログのウインドウハンドルをセーブする。ステップ１６０４において、この方法は、その基礎となるウインドウシステムファンクションＳｈｏｗＷｉｎｄｏｗを呼び出し、モードレスダイアログに関連したウインドウを隠す。ステップ１６０５において、このファンクションは、表示されたモードレスダイアログが更にあるかどうかを調べるようにチェックを行い、もしあれば、このファンクションはステップ１６０３へ戻るようにループし、さもなくば、復帰する。

４．３．６　IOLEInPlaceFrame::TranslateAccelerator

ＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒ方法は、サーバアプリケーションがそれ自身確認しないキーストロークを受け取るときに、アクセラレータキー組合せを処理できるようにする。アクセラレータキー組合せは、メニューコマンドに対するキーボードショートカットであり、これについては以下で述べる。ＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒ方法は、サーバアプリケーションメッセージポンプにおいて呼び出されるファンクションＯｂｊｅｃｔＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒによって間接的に呼び出される。コンテナーアプリケーションは、その通常のアクセラレータ処理を実行し、そしてアクセラレータが処理されたかどうかの指示を返送しなければならない。この値は、次いで、ファンクションＯｂｊｅｃｔＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒによってサーバアプリケーションへ通される。メッセージはサーバアプリケーションからコンテナーアプリケーションへ転送されているので、その基礎となるウインドウシステムは、指定されたメッセージに関連した付加的なキー状態又はメッセージ情報を保持していない。

４．４　ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔインターフェイス

テーブル５は、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔインターフェイスをリストしている。このＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔインターフェイスは、親

ウインドウと通信するためにサーバアプリケーションによって呼び出された方法を与える。又、このウインドウは、オブジェクトに対するイン・プレース「コンテナーサイト」とも称する。

テーブル５

```
interface IOLEInPlaceParent: public IOLEWindow {public:
    vitrual SCODE CanInPlaceActivate()=0;
    vitrual SCODE OnInPlaceActivate()=0;
    vitrual SCODE OnUIActivate()=0;
    vitrual SCODE OnUIDeactivate()=0;
    vitrual SCODE OnDeactivate()=0;
    vitrual SCODE ShadeBorder(LPLECT lprect.DWORD grfStatus)=0;
    vitrual SCODE GetWindowContext(IOLEInPlaceFrame*pFarme.
IOLEInPlaceUIWindow*pDoc. IOLEInPlaceUIWindow*pPane. LPRECT lprectChild
Position. HANDLE*hAccelTable)=0;
  }
```

４．４．１　IOLEInPlaceParent::CanInPlaceActivate

ＣａｎＩｎＰｌａｃｅＡｃｔｉｖａｔｅ方法は、コンテナアプリケーションがイン・プレース対話をサポートするかどうかを判断するためにサーバアプリケーションによって使用される。この方法は、選択されたコンテニーオブジェクトに代わってアクチベーションを受け入れるか又は拒絶する機会をコンテナアプリケーションに与える。この方法は、コンテナアプリケーションがイン・プレース対話を許容するかどうかの指示を返送する。

４．４．２　IOLEInPlaceParent::OnInPlaceActivate

ＯｎＩｎＰｌａｃｅＡｃｔｉｖｅ方法は、サーバアプリケーションが（アクチベーション時に）新たな複合メニューバーを形成する前に必要なオペレーションを実行する機会をコンテナアプリケーションに与えるようにサーバアプリケーションによって呼び出される。図１７は、IOLEInPlaceParent::OnInPlaceActivate方法の具現化を示すフローチャートである。この具現化において行われるオペレーションは、コンテニーオブジェクトがアクチベートされたことを指示するフラ

グをステップ１７０１においてセットすることだけである。この情報は、特定のオブジェクトの親コンテナーオブジェクトがアクチベート又はデアクチベートするように求められたときに後で使用される。このフラグは、親コンテナーアプリケーションに、それ自身のユーザインターフェイスをアクチベート又はデアクチベートするのではなく、その中に収容されたオブジェクト（ネスト状オブジェクト）をアクチベート又はデアクチベートする必要があるかどうかを知らせる。

４．４．３　IOLEInPlaceParent::OnUIActivate

ＯｎＵＩＡｃｔｉｖａｔｅ方法は、コンテニーオブジェクトをその位置においてアクチベートする準備として全てのコンテナーアプリケーションメニュー及びツールを除去する。図１８は、IOLEInPlaceParent::OnUIActivate方法の具現化を示すフローチャートである。この方法によって行われるステップは、コンテナーオブジェクトがその位置でアクチベートされたオブジェクトそれ自体であるかどうかに基づいている。ステップ１８０１において、この方法は、コンテナーオブジェクトがその位置においてアクチベートされたかどうか判断する。もしそうでなければ、この方法は、ステップ１８０２へ続き、さもなくば、ステップ１８０３へ続く。ステップ１８０２において、コンテナーオブジェクトは最上レベルのコンテナオブジェクトである（その位置でアクチベートされない）ので、この方法は、その通常の手順を用いて、コンテナーアプリケーションメニュー及び余分なツールを除去し、そして復帰する。ステップ１８０３において、コンテナーオブジェクトはコンテニーオブジェクトでもあるので、この方法は、コンテナーオブジェクトをコンテニーオブジェクトとして処理する方法をアクセスするためにオブジェクト自身のＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔを検索する。ステップ１８０４において、この方法は、コンテナーオブジェクトのアクチベート方法を呼び出してそれ自身をデアクチベートする。ステップ１８０５において、この方法は、全てのコンテナーオブジェクトの文書及び区画ウインドウレベルツールを隠す。ステップ１８０６において、この方法は、親コンテナーオブジェクトのシェードボーダー方法を呼び出して、コンテナーオブジェクトの周りからのイン・プレース対話ユーザフィードバックを除去し、そして復帰する。コンテナーオ

ブジェクトのオブジェクトウインドウは、実際には、後でデアクチベートされる（例えば、コンテニーオブジェクトがデアクチベートするときに）。

４．４．４　IOLEInPlaceParent::OnUIDeactivate

ＯｎＵＩＤｅａｃｔｉｖａｔｅ方法は、そのユーザインターフェイスリソースのデアクチベート動作の終わりにサーバアプリケーションによって呼び出され、

その親コンテナーアプリケーションがそれ自身のユーザインターフェイスをアクチベートするか又はその親コンテナーアプリケーションを呼び出して、その親コンテナーアプリケーションがそのユーザインターフェイスをアクチベートできるようにする。図１９は、IOLEInPlaceParent::OnUIDeactivate方法の具現化を示すフローチャートである。この方法は、コンテナーオブジェクトがコンテニーオブジェクトそれ自体であるか又は最上レベルのコンテナーオブジェクトであるかに基づいて２つの異なる振る舞いを与える。前者の場合に、このコンテナーがその位置でアクチベートされる新たなオブジェクトになった場合に、それ自身のユーザインターフェイスがアクチベートされ、さもなくば、コンテナーは、その親コンテナーアプリケーションに、そのユーザインターフェイスをアクチベートするように要求する。後者の場合には、コンテナーアプリケーションは、通常の手順を用いてそのユーザインターフェイスを復帰させる。ステップ１９０１において、この方法は、コンテナーアプリケーションがコンテニーオブジェクトをアクチベートしたことを指示するフラグをクリアする。ステップ１９０２において、この方法は、特定されたコンテナーオブジェクトがコンテニーオブジェクトであるかどうかを決定し、もしそうでなければ、ステップ１９０３へ続き、さもなくば、ステップ１９０５へ続く。ステップ１９０３において、この方法は、通常の手順を用いてコンテナーアプリケーションのメニュー及びそのタイトルバーをセットし、そしてステップ１９０４へと続いて、入力フォーカスを所望のウインドウへセットし、復帰する。入力フォーカスは、特定のウインドウがキーボード入力を受け取るべきときにその特定のウインドウをセットする。ステップ１９０５において、この方法は、フラグＡＢＯＵＴ＿ＴＯ＿ＡＣＴＩＶＡＴＥを検討してコンテナーオブジェクトがアクチベートされたオブジェクトになりつつあるかを

判断し、もしそうでなければ、ステップ１９０６へ続き、さもなくば、ステップ１９０７へ続く。（ＡＢＯＵＴ＿ＴＯ＿ＡＣＴＩＶＡＴＥフラグは、例えば、以下に詳細に述べるＰｒｏｃｅｓｓ＿Ｍｏｕｓｅ＿ＬＢｕｔｔｏｎＵｐファンクションにおいてコンテナーアプリケーションがユーザによって選択されたときにセットされる。）ステップ１９０６において、この方法は、コンテナーオブジェクトのIOLEInPlaceParent::OnUIDeactivate方法を呼び出して、親コンテナオブジェクトのコンテナーアプリケーションのユーザインターフェイスをアクチベートし、そして復帰する。ステップ１９０７において、この方法は、ファンクションＡｃｔｉｖａｔｅＵＩを呼び出し、コンテナーアプリケーションのユーザインターフェイスをアクチベートし、そして復帰する。

４．４．５　IOLEInPlaceParent::OnDeactivate

ＯｎＤｅａｃｔｉｖａｔｅ方法は、サーバアプリケーションによって呼び出され、それに関連したコンテナアプリケーションに、アクチベートされたコンテニーオブジェクトが完全にデアクチベートされる前にそのコンテニーオブジェクトに関連したフラグをセットするか又はいずれかのリソースを解放する機会を与える。この方法は、コンテニーオブジェクトのIOLEInPlaceObj ect::InPlaceDeactivate方法から呼び出される。

４．４．６　IOLEInPlaceParent::ShadeBorder

ＳｈａｄｅＢｏｒｄｅｒ方法は、選択されるか又は選択解除されようとしているコンテニーオブジェクトの周りに陰影付けされたパターンボーダーを描くか又はそれを除去する。陰影付けされたパターンボーダーは、コンテニーオブジェクトがその位置でアクチベートされたというユーザフィードバックを与えるのに用いられる。この方法は、ヘルパーオブジェクトリンク及び埋め込みＡＰＩファンクションＯｂｊｅｃｔＳｈａｄｅを呼び出して、適切なシェードパターンを形成することができる。この方法は、２つのパラメータ、即ちボーダーを配置しなければならないオブジェクトの周りの四角形と、１組のフラグとを取り出す。この１組のフラグは、ボーダーをオン（ＳＨＡＤＥＢＯＲＤＥＲ＿ＯＮ＝１）にすべきかオフにすべきかを指示すると共に、ボーダーをアクティブウインドウのタイ

トルバーに含まれたテキストと同じカラーで描くべき（ＳＨＡＤＥＢＯＲＤＥＲＡＣＴＩＶＥ＝１）か又はディスエイブルされたテキストと同じカラーで描くべきかを指示する。アクティブウインドウは、入力フォーカスをもつウインドウである。

４．４．７　IOLEInPlaceParent::GetWindowContext

ＧｅｔＷｉｎｄｏｗＣｏｎｔｅｘｔ方法は、特定のコンテニーオブジェクトに関連した１組のコンテナーアプリケーションインターフェイスを返送する。より

詳細には、次のパラメータを返送する。

ｐＦｒａｍｅ：これはＩＯＬＥＩｎＰｌａｃｅフレームインターフェイスに対するポインタである。

ｐＤｏｃ：これはＩＯＬＥＩｎＰｌａｃｅＵＩＷｉｎｄｏｗインターフェイスに対するポインタである。

ｐＰａｎｅ：これはＩＯＬＥＩｎＰｌａｃｅＵＩＷｉｎｄｏｗインターフェイスに対するポインタである。

ｌｐｒｅｃｔＣｈｉｌｄＰｓｏｎ：これは関連するＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔインスタンスが親ウインドウ内にオブジェクトのオブジェクトウインドウを表示する場所に対するポインタである。

ｈＡｃｃｅｌＴａｂｌｅ：これはコンテナーアプリケーションのアクセラレータテーブル（以下に述べる）に対するハンドルである。

これらの値は、アクチベーション及びデアクチベーションを交渉しそして取り扱うためにサーバアプリケーションによって使用される。この方法は、これらインターフェイスのインスタンスを形成してそれをコンテナーアプリケーションの当該フレーム、文書、区画及び親ウインドウと関連させる。

４．５　ＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイス

テーブル６は、ＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイスをリストしたものである。ＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイス方法は、収容されたオブジェクトをアクチベート及びデアクチベートするためにコンテナーアプリケーションによって呼び出される。これら方法の幾つかは、ネスト

状に収容されたオブジェクトを収容ハイアラーキによりアクセスする。他の方法は、編集メニューを表示するコンテニーオブジェクトである現在アクティブなオブジェクトのみをアクセスする。別の具現化は、このインターフェイスを２つの他のものに分割することであり、その１つは、アクティブなオブジェクトのみをアクセスすることであり、そしてもう１つは、収容ハイアラーキによりコンテニーオブジェクトをアクセスすることである。

テーブル６

```
interface IOLEInPlaceObject: public IOLEWindow {public:
virtual SCODE InPlaceDeactivate()=0;
virtual SCODE InPlaceUIDeactivate()=0;
virtual SCODE TranslareAccelerator(LPMSG lpmsg)=0;
virtual SCODE Activate(BOOL fActivate. BOOL fDocActivate)=0;
virtual SCODE ResizeBorder(RECT borderRect)=0;
virtual SCODE EnableModeless(BOOL fEnable)=0;
virtual SCODE SetVisRect(LPRECT lprect)=0;
}
```

４．５．１ IOLEInPlaceObject::InPlaceDeactivate

ＩｎＰｌａｃｅＤｅａｃｔｉｖｅ方法は、コンテナーアプリケーションによって呼び出され、「ｕｎｄｏ」オペレーションがコンテニーオブジェクトをアクセスする必要がもはやなくなった後であって且つコンテナーアプリケーションが閉じる前にコンテニーオブジェクトを完全にデアクチベートする。この方法は、コンテニーオブジェクトをその位置でアクチベートすることに関連したリソースの最終的な割り当て解除を実行する。図２０は、IOLEInPlaceObj ect::InPlaceDeactivate方法を具現化するフローチャートである。この方法は、先ず、そこに収容された（ネスト状）オブジェクトをアクチベートしたかどうかを決定し、もしそうであれば、ネスト状オブジェクトのＩｎＰｌａｃｅＤｅａｃｔｉｖｅ方法を呼び出す。さもなくば、オブジェクトはそれ自身でデアクチベートする。ステップ２００１において、この方法は、特定されたオブジェクトがコンテナーオブジェクトでもありそしてネスト状のコンテニーオブジェクトをアクチベートしてい

るかどうか判断する。もしそうであれば、この方法は、ステップ２００２へ続き、さもなくば、ステップ２００４へ続く。ステップ２００２において、この方法はアクチベートされたコンテニーオブジェクト（特定されたオブジェクトのサーバアプリケーションが既に記憶している）のＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイスを検索し、そしてステップ２００３において、その検索されたインターフェイスのIOLEInPlaceObject::InPlaceDeactivate方法を呼び出し、そして復帰する。ステップ２００４において、この方法は、特定されたオブジェクトのユーザインターフェイスがまだアクティブであるかどうかを調べるようにチ

ェックし、もしそうであれば、ステップ２００５へ続き、さもなくば、ステップ２００６へ続く。ステップ２００５において、この方法は、指定されたオブジェクトのＩｎＰｌａｃｅＵＩＤｅａｃｔｉｖａｔｅ方法を呼び出してそれ自身のユーザインターフェイスをデアクチベートし、次いで、ステップ２００６へ続く。ステップ２００６において、この方法は、サーバアプリケーションファンクションＲｅｍｏｖｅ＿Ｍｅｎｕを呼び出して、複合メニューバーからサーバアプリケーションメニューを除去する。ステップ２００７において、この方法は、特定されたオブジェクトのIOLEInPlaceFrame::RemoveMenus方法を呼び出して、親コンテナーアプリケーションがそのメニューを複合メニューバーから除去できるようにする。ステップ２００８において、この方法は、オブジェクトリンク及び埋め込みＡＰＩファンクションＯｂｊｅｃｔＤｅｓｔｒｏｙＳｈａｒｅｄＭｅｎｕを呼び出し、複合メニューバーの構造を割り当て解除する。このＯｂｊｅｃｔＤｅｓｔｒｏｙＳｈａｒｅｄＭｅｎｕファンクションは、ウインドウシステムに特定のもので、複合メニューバーに関連した構造を割り当て解除するためにその基礎となるウインドウシステムファンクションが必要とするものを呼び出す。ステップ２００９において、この方法は、その基礎となるウインドウシステムファンクションＤｅｓｔｒｏｙＭｅｎｕを呼び出し、複合メニューバー構造を割り当て解除すると共に、特定されたオブジェクトに関連したウインドウを割り当て解除する。最終的に、ステップ２０１０において、この方法は、特定されたオブジェク

トのコンテナーオブジエクトのIOEInPlaceParent::OnDeactivate方法を呼び出し、そして復帰となる。

４．５．２ IOLEInPlaceObject::InPlaceUIDeactivate

ＩｎＰｌａｃｅＵＩＤｅａｃｔｉｖａｔｅ方法は、その位置でアクチベートされた特定のオブジェクトに関連した全てのユーザインターフェイスエレメントを隠す。この方法は、複合文書内の異なるオブジェクト又はエリアのユーザ選択を処理するときにコンテナーオブジェクトによって呼び出されるか、又は特定のオブジェクトのユーザインターフェイスがまだデアクチベートされていない場合に特定のオブジェクトのＩｎＰｌａｃｅＤｅａｃｔｉｖａｔｅファンクション（図２０参照）から呼び出される。図２１は、IOLEInPlaceObj ect::InPlaceUIDeact

i

vate方法を具現化するフローチャートである。この方法は、先ず、特定のオブジェクトがコンテナーオブジェクトであってネスト状オブジェクトをアクチベートしているかどうかを決定し、もしそうならば、ネスト状オブジェクトのＩｎＰｌａｃｅＵＩＤｅａｃｔｉｖａｔｅファンクションを呼び出す。さもなくば、この方法は、それ自身のユーザインターフェイスを隠し、そのユーザインターフェイスをデアクチベートしたことをそのコンテナーアプリケーションに知らせる。ステップ２１０１において、この方法は、ネスト状コンテニーオブジェクトがその位置でアクチベートされたことを指示するフラグが真であるかどうかを決定し、もしそうであれば、ステップ２１０２へ続き、さもなくば、ステップ２１０４へ続く。ステップ２１０２において、この方法は、アクチベートされたネスト状コンテニーオブジェクトに対するＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイスを検索し、そしてステップ２１０３において、ネスト状コンテニーオブジェクトのＩｎＰｌａｃｅＤｅａｃｔｉｖａｔｅ方法を呼び出し、そして復帰となる。ステップ２１０４において、この方法は、フラグＡＢＯＵＴ＿ＴＯ＿ＡＣＴＩＶＡＴＥをクリアし、ユーザが異なるオブジェクトを選択したことを示す。ステップ２１０５において、この方法は、特定のオブジェクトのアクチベート方法を呼び出し、偽のパラメータを送って、デアクチベートする方法を要求する。こ

の方法は、親コンテナーアプリケーションのフレームウインドウに関連した全ての特定のオブジェクトのユーザインターフェイスエレメントを除去する。ステップ２１０６において、この方法は、オブジェクトリンク及び埋め込みＡＰＩファンクションＳｅｔＡｃｔｉｖｅＯｂｊｅｃｔＨｗｎｄを呼び出し、特定のオブジェクトのＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイスを、親コンテナアプリケーション文書ウインドウとの関連から除去する。これは、コンテナーアプリケーションがＭＤＩアプリケーションである場合、及びユーザが後でこの文書ウインドウを選択する場合に、特定のオブジェクトがその位置でもはや再アクチベートされないことを意味する。ステップ２１０７において、この方法は、その基礎となるウインドウシステムファンクションを用いて、親コンテナーアプリケーションの区画又は文書ウインドウに関連したサーバアプリケーションに属するユーザインターフェイスエレメントを隠す。ステップ２１０８において、この

方法は、特定のオブジェクトのIOLEInPlaceParent::ShadeBorder方法を呼び出して、デアクチベートするオブジェクトの周りから陰影付けされたボーダーパターンフィードバックを除去する。ステップ２１０９において、この方法は、その基礎となるウインドウシステムファンクションを呼び出し、特定のオブジェクトに関連したウインドウを隠す。最後に、ステップ２１１０において、この方法は、IOLEInPlaceParent::OnUIDeactivate方法を呼び出し、コンテナーアプリケーションがそれ自身のユーザインターフェイスをインストールし、そして復帰する。

４．５．３　IOLEInPlaceObject::TranslateAccelerator

ＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒ方法は、コンテナーアプリケーションがアクセラーータキーの組合せを処理する機会をもつ前にサーバアプリケーションがそれらの組合せを処理できるようにする。アクセラーータキーの組合せは、メニューコマンドに対するキーボードショートカットであり、以下に詳細に説明する。本発明の好ましい実施例では、その位置でアクチベートされるオブジェクトは、規定により、先ずアクセラレータキーの組合せを処理する。ＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒ方法は、そのメッセージポンプにおいてコンテナーアプリケーションによって呼び出される（コードテーブル９を参照された

い）。この方法によって実行される必要のある唯一のオペレーションは、特定のサーバアプリケーションのアクセラレータテーブルでその基礎となるウインドウシステムファンクションＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒを呼び出すことである。このような呼び出しは、コンテニーオブジェクトが個別の実行可能なプロセスによって具現化される場合には必要とされない。というのは、個別のプロセスは、これらキーの組合せをそれ自身のメッセージポンプにおいて受け取るが、コンテナーアプリケーションはそれらを決して受け取らないからである。この場合に、ＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒ方法は何も行わない。

４．５．４　IOLEInPlaceObject::Activate

Ａｃｔｉｖａｔｅ方法は、指定されたフラグｆＡｃｔｉｖｅが真であるか偽であるかに基づいて、親コンテナアプリケーションのフレームウインドウにインストールされたユーザインターフェイスエレメントをアクチベート又はデアクチベートする。ＭＤＩ文書ウインドウがアクチベート又はデアクチベートされたときに呼び出された場合には、この方法は、その位置でアクチベートされたオブジェクトに関連した複合メニューバーをインストールするか又は除去し、そして特定のオブジェクトがアクチベートされた場合にはその周りに陰影付けされたボーダーパターンを配する。最上レベルのフレームウインドウがアクチベート又はデアクチベートされたときに呼び出された場合には、この方法は、特定のオブジェクトがアクチベートされた場合にその周りに陰影付けされたボーダーパターンを配し、さもなくば、それを除去する。この場合に、他のユーザインターフェイスエレメントをアクチベート又はデアクチベートする必要はない。図２２は、IOLEInPlaceObject::Activate方法の具現化を示すフローチャートである。ステップ２２０１において、この方法は、最上レベルフレームウインドウのアクチベート又はデアクチベート動作の結果として呼び出されたか、或いはＭＩＤ（子）文書ウインドウのアクチベート又はデアクチベート動作の結果として呼び出されたかを決定する。ＭＩＤ文書ウインドウのアクチベート又はデアクチベート動作の結果として呼び出された場合には、この方法は、ステップ２２０２へ進み、さもなくば、ステップ２２１０へ続く。ステップ２２０２において、この方法は、特定の

オブジェクトをアクチベートすべきかどうかを決定し、もしそうでなければ、ステップ２２０３へ続き、さもなくば、ステップ２２０６へ続く。ステップ２２０３において、この方法は、親コンテナーオブジェクトのIOLEInPlaceFrame::SetMenu方法を呼び出し、その位置での特定のオブジェクトのアクチベーションに関連した複合メニューバーを除去する。ステップ２２０４において、この方法は、親コンテナーのアプリケーションフレームウインドウにインストールされたユーザインターフェイスエレメントを隠す。ステップ２２０５では、この方法は、親コンテナーオブジェクトのIOLEInPlaceParent::ShadeBorder方法を呼び出し、特定のオブジェクトの周りから陰影付けされたボーダーパターンを除去し、復帰となる。ステップ２２０６において、この方法は、親コンテナーオブジェクトのIOLEInPlaceFrame::SetMenu方法を呼び出し、複合メニューバーを関連フレームウインドウのメニューバーとしてインストールする。ステップ２２０７において、この方法は、コンテナーアプリケーションのフレームウインドウのタイトルバーをセットして、コンテナーアプリケーションが特定のオブジェクトをアクチベートしたことを指示する。ステップ２２０８において、この方法は、その基礎となるウインドウシステムファンクションを呼び出し、フレームレベルのユーザインターフェイスエレメントを表示する。ステップ２２０９において、この方法は、親コンテナーオブジェクトのIOLEInPlaceParent::ShadeBorder方法を呼び出し、特定のオブジェクトの周りに陰影付けされたボーダーパターンを描いて、その位置でアクチベートされたことを指示し、次いで、復帰となる。ステップ２２１０において、この方法は、特定のオブジェクトをアクチベートすべきであるかどうかを決定し、もしそうであれば、ステップ２２０９へ続き、さもなくば、ステップ２２１１へ続く。ステップ２２１１において、この方法は、特定のオブジェクトの周りから陰影付けされたボーダーパターンを除去し、復帰となる。

４．５．５　IOLEInPlaceObject::ResizeBorder

ＲｅｓｉｚｅＢｏｒｄｅｒ方法は、コンテナーアプリケーションによって呼び出され、サーバアプリケーションが親コンテナアプリケーションの区画又は文書ウィンドウ内に配置したユーザィンターフェイスツールをサイズ変更するように

サーバアプリケーションに要求する。この方法が呼び出されるのに応答して、サーバアプリケーションは、区画又は文書ウインドウに関連したインターフェイスインスタンスのＱｕｅｒｙＢｏｒｄｅｒＳｐａｃｅ及びＳｅｔＢｏｒｄｅｒＳｐａｃｅ方法を用いてコンテナーアプリケーションとの別のツール配置交渉ループを開始しなければならない。

４．５．６　IOLEInPlaceObject::EnableModeless

ＥｎａｂｌｅＭｏｄｅｌｅｓｓ方法は、サーバアプリケーションに対して現在表示されているモードレスダイアログをイネーブル又はディスエイブルする。典型的に、この方法は、図１６を参照して上記したIOLEInPlaceFrame::EnableModelessと同様に具現化される。

４．５．７　IOLEInPlaceObject::SetVistRect

ＳｅｔＶｉｓｔＲｅｃｔ方法は、最も内側のレベルのコンテナーオブジェクトによって呼び出され、実際に目に見えるオブジェクトの量を通信する。オブジェクトの目に見える（切り取り）四角形は、例えば、ボーダーの交渉、スクロール又はサイジングによって変更されてもよい。指定された四角形は、切り取り四角形であり、コンテニーオブジェクトのウインドウを正しい（切り取った）目に見えるサイズにサイズ変更するのはサーバアプリケーションの役目である。

４．６　その他のサーバアプリケーションファンクション

好ましい実施例において、サーバアプリケーションは、次の１組のファンクションを与える。即ち、ＡｃｔｉｖａｔｅＵＩ：ＣｒｅａｔｅＮｅｗＭｅｎｕ：ＣｒｅａｔｅＯｂｊｅｃｔＴｏｏｌＢａｒｓ：及びＲｅｍｏｖｅＭｅｎｕｓｏ。

４．６．１　ActivateUI

SCODE ActivateUI(IOLEInPlaceUIWindow*pDoc,IOLEInPlaceObject pObject)

ＡｃｔｉｖａｔｅＵＩファンクションは、指定されたコンテニーオブジェクトのユーザインターフェイスリソースのアクチベーションを制御するためにサーバアプリケーションによって具現化されるファンクションである。この高レベルファンクションは、フレーム、文書及び区画レベルのユーザインターフェイスエレ

メントをアクチベートし、オブジェクトの周りに陰影付けされたボーダーパターンを描き、そして複合メニューバーを表示する。図２３は、ＡｃｔｉｖａｔｅＵＩファンクションを具現化するフローチャートである。このファンクションは、２つのパラメータ、即ち文書インターフェイスに対するポインタと、コンテニーオブジェクトに対するポインタとを取り出す。ステップ２３０１において、このファンクションは、指定された文書ウインドウ及び指定されたコンテニーオブジェクトに対するウインドウハンドルを得る。ステップ２３０２において、このファンクションは、オブジェクトリンク及び埋め込みＡＰＩファンクションＳｅｔＡｃｔｉｖｅＯｂｊｅｃｔＨｗｎｄを呼び出し、指定された文書ウインドウの現在アクティブオブジェクトを、コンテニーオブジェクトのインターフェイスに対するポインタにセットする。これは、文書ウインドウの１つがユーザにより選択されたときに、ＭＤＩアプリケーションとして具現化されるコンテナーアプリケーションが適切なコンテニーオブジェクトをアクチベートできるようにする。ステップ２３０３では、このファンクションは、IOLEInPlaceObject::Activate方法を呼び出し、指定されたオブジェクトをアクチベートする。ステップ２３０４では、このファンクションは、その基礎となるウインドウシステムファンクションＳｈｏｗＷｉｎｄｏｗを呼び出し、コンテナアプリケーションの区画又は文書ウインドウに関連したユーザインターフェイスエレメントを表示する。ステップ２３０５において、このファンクションは、指定されたコンテニーオブジェクトを取り巻くためのボーダー又は四角形の寸法を決定し、そしてステップ２３０６において、このファンクションは、指定されたコンテニーオブジェクトのIOLEInPlaceObject::ShadeBorder方法を呼び出し、この四角形を用いて指定されたコンテニーオブジェクトの周りに陰影付けされたボーダーパターンを描く。ステップ２３０７において、このファンクションは、入力フォーカスを、指定されたコンテニーオブジェクトのオブジェクトウインドウにセットする。最終的に、ステップ２３０８において、このファンクションは、その基礎となるウインドウシステムファンクションＤｒａｗＭｅｎｕＢａ　ｒを呼び出して、複合メニューバーを再表示し、そして復帰となる。

４．６．２ CreateNewMenu

HANDLE CreateNewMenu(IOLEInPlaceFrame*pFrame)

Code Table 2

```
HANDLE CreateNewMenu (IOLEInPlaceFrame *pFrame) {
1   hmenu = CreateMenu ( );
2   pFrame -> InsertMenus (hmenu, ContrCounts);
3   lpiMenuCount[0] = ContrCount[0];
4   lpiMenuCount[2] = ContrCount[1];
5   lpiMenuCount[4] = ContrCount[2];
6   if there are Edit group menus present {
7       for each editgroupmenu {
8           heditmenu = CreateMenu ( );
9           insertion_point = lpiMenuCount[0] + lpiMenuCount[1] + 1;
10          InsertMenu (hmenu, insertion_point, MF_BYPOSITION | MF_POPUP,
11                  heditmenu, NULL);
12          lpiMenuCount[1] = lpiMenuCount[1] + 1;
13          for each editgroupmenu_item {
14              InsertMenu (heditmenu, -1, MF_BYPOSITION | MF_STRING, item_id,
15                  "string to be displayed");
16          }
17      };
18  if there are Object group menus present {
19      for each objectgroupmenu {
20          hobjmenu = CreateMenu ( );
21          insertion_point = lpiMenuCount[0] + lpiMenuCount[1] + lpiMenuCount[2]
+
22                  lpiMenuCount[3]+ 1;
23          InsertMenu (hmenu, insertion_point, MF_BYPOSITION | MF_POPUP,
24                  hobjmenu, NULL);
25          lpiMenuCount[3] = lpiMenuCount[3] + 1;
26          for each objectgroupmenu_item {
27              InsertMenu (hobjmenu, -1, MF_BYPOSITION | MF_STRING, item_id,
28                  "string to be displayed");
29          }
30      };
31  if there are Help group menus present {
32      for each helpgroupmenu {
33          hhelpmenu = CreateMenu ( );
34          InsertMenu (hmenu, -1, MF_BYPOSITION | MF_POPUP, hhelpmenu,
35                  NULL);
36          lpiMenuCount[5] = lpiMenuCount[5] + 1;
37          for each objectgroupmenu_item {
38              InsertMenu (hhelpmenu, -1, MF_BYPOSITION | MF_STRING,
item_id,
39                  "string to be displayed");
40          }
41      };
42  hSharedMenu = ObjectCreateSharedMenu (hmenu, lpiMenuCount);
43  return (hSharedMenu);
}
```

ＣｒｅａｔｅＮｅｗＭｅｎｕファンクションは、複合メニューバーの形成を管理するためにサーバアプリケーションによって具現化されるファンクションであ

る。このファンクションは、複合メニューバーに関連した構造を割り当て、そのメニューを挿入するようにコンテナーアプリケーションを要求し、そしてサーバアプリケーションメニューを挿入する。コードテーブル２は、ＣｒｅａｔｅＮｅｗＭｅｎｕファンクションの具現化を表している。ライン１において、このファンクションは、その基礎となるウインドウシステムファンクションを呼び出し、複合メニューバーに対するデータ構造を形成する。ライン２において、このファンクションは、コンテナーアプリケーションのフレームウインドウのIOLEInPlaceFrame::InsertMenusを呼び出し、コンテナアプリケーションのメニューを複合メニューバーに挿入する。ライン３ないし５において、このファンクションは、各メニューグループに対してコンテナーアプリケーションが挿入したメニューの数を追跡する。ライン６ないし１７において、サーバアプリケーションがＥｄｉｔグループメニューを有すると仮定すれば、このファンクションは、このファンクションは、各Ｅｄｉｔグループメニューを形成し、そしてそれを複合メニューバーにおける正しいスポットに挿入し、いかに多くのメニューが挿入されたかを追跡する。正しいスポットは、ライン９において、いかに多くのメニューが既に左側へ挿入されたかを決定することにより計算される。これは、Ｅｄｉｔグループに対し、コンテナーアプリケーションがコンテナーグループの一部として挿入したメニューの数と、既に挿入されているＥｄｉｔグループのメニューの数と、現在挿入に対する１との和となる。ライン１８ないし３０、３１ないし４１において、このファンクションは、各々Ｏｂｊｅｃｔグループ及びＨｅｌｐグループに属するメニューに対して同様のステップを実行する。ライン４２において、このファンクションは、オブジェクトリンク及び埋め込みＡＰＩファンクションＯｂｊｅｃｔＣｒｅａｔｅＳｈａｒｅｄＭｅｎｕを呼び出して、複合メニューバーに対するメッセージハンドリングに関連したデータ構造を形成し、そしてライン４３において、ハンドルをこの構造に返送する。

４．６．３　CreateObjectToolbars

```
void CreateObjectToolbars(IOLEInPlaceFrame*pFrame,
IOLEInPLaceUIWindow*pDoc,IOLEInPLaceUIWindow*pPane)
```

ＣｒｅａｔｅＯｂｊ ｅｃｔＴｏｏｌｂａｒｓフアンクシヨンは、サーバアプリケーションツールに必要なスペースに対しサーバアプリケーションとコンテナーアプリケーションとの間で交渉するためにサーバアプリケーションによって具現化されるファンクションである。図２４は、ＣｒｅａｔｅＯｂｊｅｃｔＴｏｏｌｂａｒｓファンクションを具現化するフローチャートである。ステップ２４０１ないし２４０８は、サーバアプリケーションに対するツールを形成するように必要に応じて何回も繰り返される。ステップ２４０１において、このファンクションは、形成すべきツールバーが更にあるかどうかを決定し、もしなければ、復帰し、さもなくば、ステップ２４０２に続く。ステップ２４０２において、このファンクションは、（サーバアプリケーションがツールをどこに配置したいかに基づいて）コンテナーアプリケーションのフレーム、文書又は区画ウインドウの１０LEInPlaceUIWindow::GetBorder方法を呼び出し、交渉を開始する。ステップ２４０３において、このファンクションは、所望のフレーム、文書又は区画ウインドウのIOLEInPlaceUIWindow::QueryBorderSpace方法を呼び出し、ＧｅｔＢｏｒｄｅｒ方法に対する以前の呼び出しにより返送された四角形の内側にボーダースペースの特定の巾を要求する。これらの方法は、以下に詳細に説明する。ステップ２４０４において、特定の巾を受け入れられない場合には、このファンクションはステップ２４０５に続き、さもなくば、復帰となる。異なる値で必要な回数だけＱｕｅｒｙＢｏｒｄｅｒＳｐａｃｅを呼び出すことにより異なる量のスペースに対して交渉するように具現化を選択することができる。ステップ２４０５において、このファンクションは、ステップ２４０３で既に交渉されているスペースでIOLEInPlaceUIWindow::SetBorderSpace方法を呼び出す。ステップ２４０６において、このファンクションは、親コンテナーアプリケーションの所望のフレーム、文書又は区画ウインドウの子ウインドウを形成する。ステップ２４０７において、このファンクションは、既に形成した子ウインドウにツールを描き、次いで、ループの始めに復帰する。

４．６．４ RemoveMenus

```
void RemoveMenus(HANDLE hSharedMenu)
```

Code Table 3

```
RemoveMenus(HANDLE hSharedMenu) {
1   menu = hSharedMenu -> menu;
2   descriptor = hSharedMenu -> descriptor;
3   for (i = descriptor[0] + 1, i <= descriptor[1]. i++) {
4       RemoveMenu (menu. heditmenu. MF_BYPOSITION)
5   }
6   for (i = descriptor[2] + 1. i <= descriptor[3]. i++ {
7       RemoveMenu (menu. hobjmenu. MF_BYPOSITION)
8   }
9   for (i = descriptor[4] + 1. i <= descriptor[5]. i++ {
10      RemoveMenu (menu. hhelpmenu. MF_BYPOSITION)
11  }
12  return ( );
}
```

ＲｅｍｏｖｅＭｅｎｕｓファンクションは、コンテニーオブジェクトのデアクチベーション時に複合メニューバーからサーバアプリケーションのメニューを除去する。この方法は、コンテニーオブジェクトのＩｎＰｌａｃｅＤｅａｃｔｉｖａｔｅ方法から呼び出される。コードテーブル３は、ＲｅｍｏｖｅＭｅｎｕｓファンクションの具現化を示す。このファンクションは、各メニューグループ内に多数のメニューを含む共用メニュー記述子（図２５参照）に記憶された情報を用いて、その基礎となるウインドウシステムファンクションＲｅｍｏｖｅＭｅｎｕを呼び出すことにより全てのサーバアプリケーションメニューを除去する。ライン３ないし５は、Ｅｄｉｔグループに属するメニューを除去し、ライン６ないし８は、Ｏｂｊｅｃｔグループに属ずるメニューを除去し、そしてライン９ないし１１は、Ｈｅｌｐグループに属ずるメニューを除去する。

４．７　オブジェクトリンク及び埋め込みＡＰＩのヘルパーファンクション

インターフェイス定義に加えて、オブジェクトリンク及び埋め込みＡＰＩは、コンテナー及びサーバアプリケーションによって使用されるべき１組のヘルパーファンクションを与える。これらのファンクションは、ＳｅｔＡｃｔｉｖｃＯｂｊｃｃｔＨｗｎｄ；ＧｅｔＡｃｔｉｖｅＯｂｊｅｃｔＨｗｎｄ；ＯｂｊｅｃｔＣｒｃａｔｃＳｈａｒｅｄＭｅｎｕ；ＯｂｊｅｃｔＤｅｓｔｒｏｙＳｈａｒｅｄＭｅｎｕ：ＯｂｊｅｃｔＳｈａｄｅ；及びＯｂｊｅｃｔＳｅｔＭｅｎｕを含む。

４．７．１　SetActiveObjectHwnd

void SetActiveObjectHwnd(HWND hwndDOC,

IOLEInPlaceObject*pObject)

ＳｅｔＡｃｔｉｖｅＯｂｊｅｃｔＨｗｎｄファンクションは、ＭＤＩアプリケーションの現在選択されたオブジェクトをセットする。ＭＤＩコンテナーアプリケーションの各ＭＤＩ（文書）ウインドウには、そのＭＤＩウインドウが最後に入力フォーカスを有したときに表示されたその位置でアクチベートされたオブジェクトに対応するオブジェクトインターフェイスが関連される。ＭＤＩウインドウ内からオブジェクトがその位置でアクチベートされていない場合には、それに関連したオブジェクトインターフェイスはＮＵＬＬである。このメカニズムは、ＭＤＩウインドウが後で入力フォーカスを受け取るときに、例えば、ユーザがＭＤＩウインドウのタイトルバーにおいてマウスでクリックするときに、適切なコンテニーオブジェクトをアクチベートするようにＭＤＩウインドウをイネーブルする。（ユーザが他のＭＤＩウインドウのタイトルバーにおいてクリックするときは、第１のウインドウに関連したいかなるイン・プレース対話も表示から消失する。）ＳｅｔＡｃｔｉｖｅＯｂｊｅｃｔＨｗｎｄファンクションは、２つのパラメータ、即ちＭＤＩ（文書）ウインドウのウインドウハンドルと、その位置で現在アクチベートされているオブジェクトのＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイスとを取り上げる。ウインドウハンドルをオブジェクトインターフェイスに関連させる方法は多数あることが当業者に明らかであろう。１つの実施例において、このファンクションは、オブジェクトインターフェイスを、その基礎となるウインドウシステムファンクションを用いて文書ウインドウの特性として記憶する。この具現化は、システムにおいてアクティブな各ＭＤＩウインドウごとに記憶装置を必要とすることに注意されたい。別の解決策は、現在選択されるオブジェクトを追跡する方法を文書、区画及びフレームウインドウインターフェイスに追加する。

４．７．２　GetActiveObjectHwnd

HWND GetActiveObjectHwnd(HWND hwndDOC)

ＧｅｔＡｃｔｉｖｅＯｂｊｅｃｔＨｗｎｄファンクションは、ＭＤＩウインド

ウが入力フォーカスを受け取るときにその位置でアクチベートされるべきコンテニーオブジェクトを検索する。このファンクションは、ＳｅｔＡｃｔｉｖｅＯｂｊｅｃｔＨｗｎｄファンクションを用いて既に記憶されているオブジェクトインターフェイスを返送する。

４７．３ ObjectCreateSharedMenu

HANDLE ObjectCreateSharedMenu(HMENU hMenuCombined.

UINT lpiMenuCounts)

Code Table 4

```
HANDLE ObjectCreateSharedMenu (hmenu, lpiMenuCount) {
1   hSharedMenu = AllocateSharedMenuHandle();
2   hSharedMenu -> menu = hmenu;
3   for (i = 0. i < number_menu_groups; i ++) {
4       if (i > 0) {
5           hSharedMenu -> descriptor[i].count =
6                   hSharedMenu -> descriptor[i - 1].count + lpiMenuCount[i];
7           if i is even
8               hSharedMenu -> descriptor[i].function = Id_Container
9           else hSharedMenu -> descriptor[i].function = Id_Object;
10      }
11      else {
12          hSharedMenu -> descriptor[0].count = lpiMenuCount[0];
13          hSharedMenu -> descriptor[0].function = Id_Container;
14      };
15  }
16  return (hSharedMenu);
}
```

ＯｂｊｅｃｔＣｒｅａｔｅＳｈａｒｅｄＭｅｎｕファンクションは、その位置でアクチベートされたオブジェクトの複合メニューバーに関連した共用メニューデータ構造を形成する。このファンクションは、コンテニーオブジェクトがアクチベートされたときにＣｒｅａｔｅＮｅｗＭｅｎｕファンクションから呼び出される。コードテーブル４は、ＯｂｊｅｃｔＣｒｅａｔｅＳｈａｒｅｄＭｅｎｕファンクションの具現化を示す。このファンクションは、２つのパラメータ、即ち複合メニューバーに対ずるハンドルと、各メニューグループにおけるメニューの

数を含むメニュカウントのアレーとを取り上げる。このファンクションは、新たに形成された共用メニューデータ造に対するハンドルを返送する。ライン１にお

いて、このファンクションは、共用メニューデータ構造に必要なメモリを割り当てる。ライン２において、このファンクションは、このデータ構造における複合メニューバーに対するハンドルをセーブする。ライン３ないし１５では、このファンクションは、メニューカウントアレーに記憶されたファンクションに基づいて共用メニュー記述子を設定する。この記述子は、メニューコマンドを受け取るときにウインドウ手順によって使用され、メニューコマンドをコンテナーアプリケーションに送るべきかサーバアプリケーションに送るべきかを決定する。

この情報を維持する種々の方法があることが当業者に明らかであろうが、１つの実施例では、記述子は、各インデックスにおいて、そのインデックスに関連したメニューグループに含まれた最後のメニューの数を記憶する。（メニューは、左側において１から番号付けされる。）又、メニューグループがコンテナーアプリケーションに属するかサーバアプリケーションに属するかの指示も各インデックスに記憶される。又、どのメニューグループに対してどのアプリケーションに通知すべきかを指示する付加的なパラメータを通すことにより、メニューグループ分け機構もサポートできることが当業者に明らかであろう。この記述子構成を用いて、ウインドウ手順は、メニュー項目選択を含むまでメニューの数をカウントしそしてそのメニュー数を記述子の値と比較して正しいインデックスを見つけることにより、特定のメニュー項目選択がどのインデックス内に入るかを決定することができる。いったんインデックスが決定されると、ウインドウ手順は、コンテナーアプリケーションファンクションを呼び出すべきか又はサーバアプリケーションファンクションを呼び出すべきかを指定する指示子を検索することができる。この手順は、コードテーブル５を参照して以下に詳細に述べる。

図２５は、図４について述べた例に対応する共用メニューデータ構造のブロック図である。この共用メニューデータ構造は、複合メニューバーのポインタ２５０２と、各メニューグループに対するメニューカウントを含む記述子２５０５とで構成される。複合メニューバー２５０３は、コンテナー及びサーバアプリケーション２５０４からのメニューを含む。記述子２５０５の各エレメントは、カウントフィールド２５０６及びファンクションフィールド２５０７を有する。カウ

ントフィールド２５０６は、メニューグループ内の最後のメニューの数（左からスタートして）を指示する。例えば、第２のメニューグループは、Ｅｄｉｔグループであり、１つのメニューしか含まない。このメニュー２５０３は、複合メニューバーにおいて左から２番目のメニューであり、それ故、カウントフィールド２５０９は、数２を含む。別の例として、４番目のメニューグループは、Ｏｂｊｅｃｔグループである。このグループは、サーバアプリケーション２５１０からの５つのメニューを含む。それ故、このメニューグループ２５１１のカウントは数７を含む。というのは、７番目のメニューが、Ｏｂｊｅｃｔグループにおける最後のメニューであるＭａｃｒｏメニューだからである。

４．７．４　ObjectDestroySharedMenu

void ObjectDestroySharedMenu(HMENUhMenuCombined)

ＯｂｊｅｃｔＤｅｓｔｒｏｙＳｈａｒｅｄＭｅｎｕファンクションは、ＯｂｊｅｃｔＣｒｅａｔｅＳｈａｒｅｄＭｅｎｕへの以前の呼び出しにおいて形成された共用メニューデータ構造を破壊する。このファンクションは、コンテナー及びサーバアプリケーションがそれらのメニューを複合メニューバーから除去した後に、アクチベートされたコンテニーオブジェクトのIOLEInPlaceObject::InPlace Deactivate方法から呼び出される。

４．７．５　ObjectShade

voidObjectShade(HWND hwndParent,LPRECT lprc.DWORD grfState)

ＯｂｊｅｃｔＳｈａｄｅファンクションは、オブジェクトリンク及び埋め込みＡＰＩによって与えられ、その位置でアクチベートされたオブジェクトの周りに配置される陰影付けされたボーダーパターンを形成する。ｈｗｎｄＰａｒｅｎｔパラメータは、アクチベートされた（又はアクチベートされるべき）オブジェクトのＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔインターフェイスに関連したウインドウハンドルである。ｌｐｒｃパラメータは、パターンが配置されるところの親ウインドウ座標における四角形である。ｇｒｆＳｔａｔｅフラグは、ＩＯＬＥＩｎＰｌａｃｅParent::ShadeBorder方法について述べたものと同じであり、ＳＨＡＤＥＢＯＲＤＥＲ＿ＯＮ及びＳＨＡＤＥＢＯＲＤＥＲ＿ＡＣＴＩＶＥを含む。

４．７．６ ObjectSetMenuDescriptor

SCODE ObjectSetMenu(HWND hwndFrame,HOLEMENU hMenuCombind,
HWND hwndObject)

ＯｂｊｅｃｔＳｅｔＭｅｎｕＤｅｓｃｒｉｐｔｏｒファンクションは、複合メニューバーに対するメッセージハンドラーを設定又は除去する。このファンクションは、アクチベートするコンテニーオブジェクトの関連コンテナーオブジェクトのIOLEInPlaceFrame::SetMenu方法によって呼び出される。図２６は、ＯｂｊｅｃｔＳｅｔＭｅｎｕＤｅｓｃｒｉｐｔｏｒファンクションの具現化を示すフローチャートである。これは、３つのパラメータ、即ちコンテナーアプリケーションに関連したフレームウインドウのウインドウハンドル：Ｏｂｊ ｅｃｔＣｒｅａｔｅＳｈａｒｅｄＭｅｎｕファンクションによって返送された共用メニューデータ構造に対するハンドル：及び現在その位置でアクチベートされるべきオブジェクトのウインドウハンドルを取り上げる。共用メニュー構造に対するハンドルがナルである場合には、このファンクションは、複合メニューバーに対するメッセージハンドラーを除去し、さもなくば、メッセージハンドラーを設定する。ステップ２６０１において、このファンクションは、指定された共用メニューデータ構造に対するハンドルがナルであるかどうか決定し、もしそうであれば、ステップ２６０２へ続き、さもなくば、ステップ２６０３へ続く。ステップ２６０２において、このファンクションは、その基礎となるウインドウシステムファンクションＳｅｔＷｉｎｄｏｗＬｏｎｇを呼び出し、コンテナーアプリケーションに既に関連されている特殊なメッセージハンドラーを除去する。ステップ２６０３において、このファンクションは、複合メニューに対して既に設定されている特性を除去し、次いで、復帰する。ステップ２６０４において、このファンクションは、特殊なメッセージハンドラーによっ後で使用されるべき共用メニューデータ構造を記憶するための特性をフレームウインドウにセットする。ステップ２６０５において、このファンクションは、アクチベートするオブジェクトのウインドウハンドルに対応する別の特性をフレームウインドウにセットする。ステップ２６０６において、このファンクションは、その基礎となるウインドウシステムファンクションＳｅｔＷｉｎｄｏｗＬｏｎｇを用いて、親コンテナーアプリケー

シ

ョンのフレームウインドウに対する新たなウインドウ手順として特殊なメッセージハンドラーをインストールする。古いウインドウ手順は、特性Ｏｌｄ＿Ｆｉｌｔｅｒに後で使用するためにセーブされる。（例えば、以下に詳細に述べるコードテーブル５を参照されたい。）次いで、ファンクションは復帰となる。

５．イン・プレース対話ＡＰＩの使用

イン・プレース対話をサポートするオブジェクトリンク及び埋め込みＡＰＩファンクションは、次のことを行うために呼び出される。

・ＳＤＩ又はＭＤＩコンテナーアプリケーション内の場所でオブジェクトをアクチベートする。

・ユーザがコンテナーアプリケーションの複合メニューバーからメニー項目を選択するときにメッセージを処理する。

・ユーザが異なるオブジェクトをアクチベートするように選択するか又は最上レベルのコンテナアプリケーションをその通常処理に回復するよう選択するときにその位置でアクチベートされたオブジェクトのユーザインターフェイスリソースをデアクチベートする。

・コンテナーアプリケーションがイン・プレース対話リソースをもはや必要としないときにサーバアプリケーションに対してそれらリソースをデアクチベートする。

・サーバアプリケーションがダイアログを表示しそしてコンテナーアプリケーションがモードレスダイアログを現在表示する（又はその逆である）ときにモードレスダイアログをイネーブル及びディスエイブルする。

・アクセラレータキー組合せをコンテナーアプリケーションとサーバアプリケーションとの間に分配するようにこれら組合せを処理する。

５．１　その位置でアクチベーションする手順

上記したように、オブジェクトが文書にリンクされるか又は埋め込まれると、ユーザは、オブジェクトを選択し、そしてその選択されたオブジェクトに基づいてあるアクションを実行するよう要求する。図３及び４の例に戻ると、ユーザが

スプレッドシートオブジェクト３０５をその位置でアクチベートするように望む場合には、ユーザは、オブジェクトプレゼンテーションフォーマットにおいてマウス入力装置で２回クリックするか、又はコンテナーアプリケーションメニューを用いてオブジェクトにおけるアクションを選択することができる。図８は、ユーザがメニューを使用してスプレッドシートオブジェクト３０５をその位置でアクチベートできるようにする１つの方法を示している。ユーザがコンテナーアプリケーション（ワードプロセスアプリケーション）のＥｄｉｔメニュー８０２からメニュー項目「Ｅｘｃｅｌ　Ｗｏｒｋｓｈｅｅｔ　Ｏｂｊｅｃｔ」８０３を選択し、そして「Ｅｘｃｅｌ　Ｗｏｒｋｓｈｅｅｔ　ＯｂｊｅｃＵサブメニュー８０４からいずれかのアクションを選択するときには、ワードプロセスアプリケーションは、スプレッドシートアプリケーションを呼び出して、スプレッドシートオブジェクトをその位置でアクチベートする。

スプレッドシートオブジェクト３０５をアクチベートするプロセスは、多数のステップで行われる。第１に、ワードプロセスアプリケーションのフレームウインドウに対するウインドウ手順が、その基礎となるウインドウシステムにより、オブジェクトアクションサブメニュー８０４におけるメニュー項目のユーザ選択に応答して呼び出される。（例えば、図１４を参照されたい。）第２に、メニュー事象を受け取ると、ウインドウ手順は、ファンクションＰｒｏｃｅｓｓ＿Ｏｂｊｅｃｔ＿Ａｃｔｉｖａｔｉｏｎを呼び出す。（例えば、ステップ１４０７、１４１１及び１４１３を参照されたい。）第３に、ファンクションＰｒｏｃｅｓｓ＿Ｏｂｊｅｃｔ＿Ａｃｔｉｖａｔｉｏｎは、オブジェクトリンク及び埋め込みＡＰＩのファンクションＯｂｊｅｃｔＬｏａｄを用いてスプレッドシートオブジェクト３０５に対するデータをロードする。最後に、ファンクションＰｒｏｃｅｓｓ＿Ｏｂｊｅｃｔ＿Ａｃｔｉｖａｔｉｏｎプログラムは、スプレッドシートオブジェクト３０５のＤｏＶｅｒｂ方法を呼び出して、選択されたアクションを実行するようにスプレッドシートアプリケーションに要求する。

図２７は、ファンクションＰｒｏｃｅｓｓＯｂｊｅｃｔＡｃｔｉｖａｔｉｏｎを具現化するフローチャートある。このファンクションは、選択されたオブジェ

クトをロードし、そしてそのＤｏＶｅｒｂ方法を呼び出して、選択されたアクションを実行する。ステップ２７０１において、このファンクションは、オブジェクトリンク及び埋め込みＡＰＩのファンクション０ｂｊｅｃｔＬｏａｄを

呼び出し、オブジェクトの記憶に対するポインタと、それがＩＯＬＥＯｂｊｅｃｔインターフェイスを希望するという指示とをそれに通す。ファンクションＯｂｊｅｃｔＬｏａｄは、ロードされたオブジェクトのＩＯＬＥＯｂｊｅｃｔインターフェイスに対するポインタを返送する。ステップ２７０２において、このファンクションは、コンテニーオブジェクトのＳｅｔＣｌｉｅｎｔＳｉｔｅ方法を呼び出して、その関連する親コンテニーオブジェクトインターフェイス（ｐｃｌｉｅｎｔｓｉｔｅ）に対するポインタをコンテニーオブジェクトに手渡す。ステップ２７０３において、このファンクションは、ロードされたオブジェクトのＩＯＬＥＯｂｊｅｃｔ：：ＤｏＶｅｒｂ方法を呼び出し、選択されたアクション、そのアクションがダブルクリックによって選択されたかどうかの指示、及びオブジェクトに対して既に形成されているＩＯＬＥＣｌｉｅｎｔＳｉｔｅインターフェイスをそこに通す。次いで、このファンクションは復帰となる。

図２８は、オブジェクトリンク及び埋め込みＡＰＩファンクションＯｂｊｅｃｔＬｏａｄの具現化を示すフローチャートである。このファンクションは、オブジェクトのメモリ内インスタンスを形成し、将来の対話についてサーバアプリケーションを準備し、そして指定されたインターフェイスに対するポインタを返送する。このファンクションは、３つのパラメータ、オブジェクトデータがロードされるべき記憶装置に対するポインタ、発呼者が返送を望むインターフェイスの指示、及びオブジェクトのメモリ内インスタンスに対する返送ポインタを取り上げる。ステップ２８０１において、このファンクションは、ＣＬＡＳＳ＿ＩＤを指定の記憶装置から検索する。ステップ２８０２において、このファンクションは、検索されたＣＬＡＳＳ＿ＩＤを用いて、この形式のオブジェクトのメモリ内インスタンスを形成するためのコードを位置決めする。本発明の好ましい実施例においては、ＩＯＬＥＣｒｅａｔｅインターフェイスが各サーバアプリケーションによって与えられ、それが具現化するオブジェクトのメモリ内インスタンスを

形成する。ステップ２８０３において、このファンクションは、ＩＯＬＥＣｒｅａｔｅ：：ＣｒｅａｔｅＩｎｓｔａｎｃｅ方法を呼び出し、オブジェクトに対するメモリ内構造を形成し、そしてオブジェクトに対する永続的記憶装置をアクセスするところのＩＰｅｒｓｉｓｔＳｔｏｒａｇｅインターフェイスに対するポインタを返送する。ステップ２８０４において、このファンクションは、ＩＰｅｒｓｉｓｔＳｔｏｒａｇｅ：：Ｌｏａｄ方法を呼び出し、これは、指定された記憶装置からオブジェクトデータをロードする。ステップ２８０５において、このファンクションは、ＩＰｅｒｓｉｓｔＳｔｏｒａｇｅ：：ＱｕｅｒｙＩｎｔｅｒｆａｃｅ方法を呼び出し、指定のインターフェイスを検索すると共に、その検索されたインターフェイスを返送する。

図２９は、ＩＯＬＥＯｂｊｅｃｔ：：ＤｏＶｅｒｂ方法の典型的な具現化を示すフローチャートである。この方法は、コンテニーオブジェクトと対話するための主たる方法である。この方法は、コンテナーアプリケーションと交渉して、イン・プレース対話を行うと共に、サーバアプリケーションのユーザインターフェイスをアクチベートさせる。この方法は、４つのパラメータ、即ちユーザが選択したアクション；ユーザがアクションを選択したときにコンテナーアプリケーションのウインドウ手順によって受け取られるメッセージ構造に対するポインタ；オブジェクトのＩＯＬＥＣｌｉｅｎｔＳｉｔｅインターフェイスに対するポインタ：及び例えば、サーバアプリケーションが呼び出し時に入力フォーカスを得るべきかどうかについてｖｅｒｂ．の実行を制御する１組のフラグを取り上げる。ステップ２９０１において、この方法は、ＩＯＬＥＣｌｉｅｎｔＳｉｔｅ：：ＱｕｅｒｙＩｎｔｅｒｆａｃｅ方法を呼び出して、特定のオブジェクトに対するＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔインターフェイスを得る。ステップ２９０２において、この方法は、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ：：ＣａｎＩｎＰｌａｃｅＡｃｔｉｖａｔｅ方法を呼び出して、コンテナーアプリケーションがイン・プレース対話をサポートするかどうかを決定する。ステップ２９０３において、コンテナーアプリケーションがイン・プレース対話をサポートしない場合には、この方法はステップ２９０４に続き、さもなくば、この方法はステップ２９

０６に続く。ステップ２９０４において、この方法は、サーバアプリケーションのフレームウインドウを形成し表示する。というのは、コンテナーアプリケーションがイン・プレース対話を実行できないからである。ステップ２９０５において、この方法は、指定されたアクションの通常の処理を続け、復帰する。ステップ２９０６において、この方法は、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ：：ＧｅｔＷｉｎｄｏｗＣｏｎｔｅｘｔ方法を呼び出し、コンテナーアプリケーションに関連したインターフェイスを得る。ステップ２９０７において、この方法は、特定のオブジェクトでイン・プレース対話をサポートするために形成する必要のあるオブジェクトウインドウのサイズを計算する。ステップ２９０８において、この方法は、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ：：ＧｅｔＷｉｎｄｏｗＣｏｎｔｅｘｔ方法により返送されたエリアがスケーリング又はクリッピングを必要とするかどうか及び特定のオブジェクトがこれをサートトできるかどうかを決定する。必要とされるサイズをサポートできる場合には、この方法はステップ２９０９に続き、さもなくば、この方法は、イン・プレース対話を放棄してステップ２９０９へ続く。ステップ２９０９において、この方法は、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔインターフェイスに対応するウインドウに対するウインドウハンドルを検索する。ステップ２９１０において、この方法は、オブジェクトルートウインドウとして使用されるべきＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔインターフェイスに対応するウインドウの子として新たなウインドウを形成する。（例えば、図１３の項目１３０９を参照されたい。）ステップ２９１１において、この方法は、特定のオブジェクトに対するユーザインターフェイスリソースがまだ使用できるかどうか、即ち割り当てられているがまだ割り当て解除されていないかどうかを決定する。これらリソースが使用できる場合には、この方法はステップ２９１３へ続き、さもなくば、この方法はステップ２９１２へ続く。ステップ２９１２において、この方法は、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ：：ＯｎＵＩＡｃｔｉｖａｔｅ方法を呼び出し、コンテナーアプリケーションが、特定のオブジェクトをその位置でアクチベートする準備においてそのユーザインターフェイスリソースを除去できるようにする。ステップ２９１３において、この方法

は、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ：：ＯｎＩｎＰｌａｃｅＡｃｔｉｖａｔｅ方法を呼び出し、コンテナーアプリケーションが、ネスト状オブジェクトをその位置でアクチベートしたことを記録できるようにする。ステップ２９１４において、この方法は、ファンクションＣｒｅａｔｅＮｅｗＭｅｎｕを呼び出し、新たな複合メニューバーを形成する（コードテーブル２を参照）。ステップ２９１５において、この方法は、特定のオブジェクトが付加的なユーザインターフェイスツールのアクチベーションを必要とするかどうかを決定し、もしそうであれば、ステップ２９１６へ続き、さもなくば、ステップ２９１７へ続く。ステップ２９１６において、この方法は、ファンクションＣｒｅａｔｅＯｂｊｅｃｔＴｏｏｌｂａｒｓを呼び出し、特定のオブジェクトの付加的なユーザインターフェイスツールの位置を交渉しそしてそれを配置する（図２４を参照）。ステップ２９１７において、この方法は、ファンクションＡｃｔｉｖａｔｅＵＩを呼び出し、これは、特定のオブジェクトの全てのユーザインターフェイスリソースを表示させ（図２３を参照）そして復帰となる。

５．１．１　多文書インターフェイスアプリケーション内の位置でのアクチベーション

オブジェクトがＳＤＩコンテナーアプリケーションからアクチベートされたと仮定してコンテニーオブジェクトのアクチベーションについて上記した。一方、オブジェクトがＭＤＩコンテナーアプリケーション（このアプリケーションは、定義により、多数の複合文書と同時に対話できる）内でアクチベートされる場合には、オブジェクトを含む文書（ＭＤＩ）ウインドウがアクチベート又はデアクチベートされるときにアクチベーション及びデアクチベーションが生じる。文書ウインドウに対するウインドウ手順は、ユーザがウインドウを選択する（例えば文書ウインドウのタイトルバーにおいてクリックすることにより）ときに、その基礎となるウインドウシステムからアクチベーションメッセージを受け取る。文書ウインドウに対するウインドウ手順は、ユーザが次いで異なるウインドウを選択するときにデアクチベーションメッセージを受け取る。これらのメッセージに応答して、文書ウインドウに対するウインドウ手順は、ファンクション（例えば

Ｐｒｏｃｅｓｓ＿Ａｃｔｉｖａｔｉｏｎ＿Ｍｅｓｓａｇｅ）を呼び出し、文書ウインドウ及びその中に含まれたアクチーｔｏトされたオブジェクトのアクチベーション及びデアクチベーションを実行する。

図３０は、アクチベーション及びデアクチベーションメッセージを処理するためにＭＤＩ文書ウインドウのウインドウ手順によって呼び出されたファンクションＰｒｏｃｅｓｓ＿Ａｃｔｉｖａｔｉｏｎ＿Ｍｅｓｓａｇｅの具現化を示すフローチャートである。１つの実施例において、文書ウインドウハンドルがファンクションに対するパラメータとして通される。このファンクションは、ウインドウが最後にアクティブとなったときに既にその位置でアクチベートされたオブジェクトをウインドウが含むかどうかを決定する。もしそうであれば、ファンクションは、そのオブジェクトをアクチベート又はデアクチベートし、さもなくば、文書ウインドウをその通常の形態でアクチベート又はデアクチベートする。ステップ３００２において、このファンクションは、その通常のウインドウデアクチベーション手順を実行し、そして復帰となる。ステップ３００１において、このファンクションは、ファンクションＧｅｔＡｃｔｉｖｅＯｂｊｅｃｔＨｗｎｄを呼び出すことにより、既にアクティブな収容オブジェクトがもしあればそれに対しＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔのオブジェクトインターフェイスを検索する。ステップ３００２において、このファンクションは、オブジェクトインターフェイスがナルであるかどうか決定する。もしナルであれば、既にアクティブな収容オブジェクトは存在せず、ファンクションはステップ３００３へ続き、さもなくば、ファンクションはステップ３００４へ続く。ステップ３００３では、このファンクションは、文書ウインドウツール及びメニューを設定するといったその通常のウインドウアクチベーション又はデアクチベーション手順を実行し、そして復帰となる。ステップ３００４では、このファンクションは、指定されたフラグＦＡｃｔｉｖｅが真であるかどうか決定する。フラグＦＡｃｔｉｖｅが真である場合は、既にアクティブな収容オブジェクトをアクチベートすべきであり、このファンクションはステップ３００５に続き、さもなくば、オブジェクトをデアクチベートすべきであり、ファンクションはステップ３００６に続く。ステッ

プ３００５では、このファンクションは、検索されたインターフェイスのＩＯＬＥＩｎｌａｃｅＯｂｊｅｃｔ：：Ａｃｔｉｖａｔｅ方法を呼び出して、既にアクチベートされたオブジェクトがそれ自体アクチベートするように要求し、そして復帰となる。ステップ３００６では、このファンクションは、検索されたインターフェイスのＩＯＬＥｌｎＰｌａｃｅＯｂｊｅｃｔ：：Ａｃｔｉｖａｔｅ方法を呼び出して、既にアクチベートされたオブジェクトがそれ自体デアクチベートするように要求し、そして復帰となる。

５．２　プルダウンメニューメッセージハンドリングのユーザ選択

ユーザがオブジェクトをその位置でアクチベートした後に、ユーザは、コンテナーアプリケーションのメニューバー（これは複合メニューバーである）を介してアクションを選択することによりコンテナーアプリケーション内のオブジェクトと対話する。メニューのあるものはサーバアプリケーションに属しておりそして他のメニューはコンテナーアプリケーションに属しているので、コンテナーアプリケーションのフレームウインドウのウインドウ手順は、メニュー入力事象をコンテナーアプリケーション内のファンクションに送信すべきかサーバアプリケーション内のファンクションに送信すべきか判断しなければならない。このために、オブジェクトリンク及び埋め込みＡＰＩのファンクションＯｂｊｅｃｔＳｅｔＭｅｎｕＤｅｓｃｒｉｐｔｏｒがサーバアプリケーションによって呼び出されたときにこのファンクションによって特殊なメッセージハンドラーがインストールされ、新たに形成された複合メニューバーをインストールする。この特殊なメッセージハンドラーは、これがインストールされると、コンテナーアプリケーションのフレームウインドウに対する新たなウインドウ手順となる。従って、そのフレームウインドウに対応するコンテナーアプリケーションによって受け取られた全てのメッセージは、その後に、先ず、特殊なメッセージハンドラーへ送られる。この特殊なメッセージハンドラーは、次いで、受け取ったメッセージ事象をどのアプリケーションへ送るべきかを判断する。

Code Table 5

```
InPlaceWndProc ( hwnd, message, wparam, lparam ) {
1 /* wparam = item id of menu item selected ;      */
2 /*  hiword (lparam) = hmenu containing the item */
3   switch (message) {
4       case WM_COMMAND:
5           SetFocus (Old_Focus);
6           if (saveMenuRoutine == Id_Object ) {
7               hwndObj = GetProp(hwndFrame, "InPlaceObject");
8               PostMessage(hwndObj, message, wparam, lparam);
9           }
10          else {
11              Old_Filter = GetProp(hwndFrame, "OldFilter");
12              call Old_Filter (hwndFrame, message, wparam, lparam);
```

```
13          };
14      break;
15      case WM_INITMENUPOPOP.
16           WM_ENTERIDLE.
17           WM_MEASUREITEM.
18           WM_DRAWITEM:
19          if (saveMenuRoutine = = Id_Object ) {
20              hwndObj = GetProp(hwndFrame. "InPlaceObject");
21              PostMessage(hwndObj. message. wparam. lparam);
22          }
23          else {
24              Old_Filter = GetProp(hwndFrame. "OldFilter");
25              call Old_Filter (hwndFrame. message. wparam. lparam);
26          };
27          break:
28      case WM_MENUSELECT:
29          hSharedMenu = GetProp(hwndFrame. "SharedMenu");
30          CombinedMenu = hSharedMenu -> menu:
31          count = 0;
32          for (current = 1st menu entry of CombinedMenu: current < = last menu entry
33                    of Combined Menu: current ++) {
34              if (current -> item_id = = hiword (lparam))
35                  found = true;
36              count = count + 1;
37          }
38          if (found)
39              GetFocus(Old_Focus):
40              SetFocus(hwndFrame):
41              descr = hSharedMenu -> descriptor;
42              saveMenuRoutine = NULL;
43              for (d = 0; d <= sizeof (descr); d++ ) {
44                  if (d = 0) and (count <= descr [d].count)
45                      saveMenuRoutine = descr[d].function:
46                  else if (count > descr[d - 1].count) and (count <= descr [d].count)
47                      saveMenuRoutine = descr[d].function;
48              }
49          break;
50      default:
51          Old_Filter = GetProp(hwndFrame. "OldFilter");
52          call Old_Filter (hwndFrame, message, wparam, lparam);
53          break:
54  }
55  return ( );
}
```

コードテーブル５は、オブジェクトリンク及び埋め込みＡＰＩにより与えられる特殊なメッセージハンドラーの具現化を示している。ライン３ないし５４は、受け取ったメッセージの形式に基づくケーステートメントを具現化ずる。ライン２８ないし４９は、ハンドラーがその基礎となるウインドウシステムからメニューコマンドメッセージを受け取ったときにメッセージを適切に送れるようにする初期化を与える。ライン４ないし２７及びライン５０ないし５３は、メニュー

コマンドメッセージが受け取られたときに基本的なルートメカニズムを与える。ライン２９において、共用メニューデータ構造に対するハンドルは、コンテナーアプリケーションのフレームウインドウの特性から検索される。ライン３２ないし３７において、ハンドラーは、複合メニューバーのエントリーをウォークし、その入力メッセージで受け取られたメニューとの一致を試みる。このプロセスにおいて、遭遇するメニューの数を追跡する（ライン３６）。ループを出たときにメニューが見つかった場合には、変数カウントは、左から始めて選択されたメニューの数を表す。このカウントは、次いで、ライン４３ないし４８に含まれたループに使用され、メニューがどの記述子エレメントに属するかを決定する。より詳細には、記述子の各インデックスに記憶された値がチェックされ、そしてメニューの数がその値以下であって且つ左側のインデックスにおいて記述子に記憶された値より大きい場合には、正しい記述子エレメントが位置決めされる。正しいエレメントが分かると、ハンドラーは、コンテナーアプリケーションファンクションを呼び出さねばならないかサーバアプリケーションファンクションを呼び出さねばならないかに対応する指示子を検索する。この指示子は、次いで、ライン４５及び４７において変数ｓａｖｅＭｅｎｕＲｏｕｔｉｎｅにセーブされる。メニューのニューモニックス（mneumonics）（システムメニューキーシーケンスを含む）を適切に処理するためのハンドラーとして、このハンドラーは、メニューコマンドメッセージに応答するときに入力フォーカスをフレームウインドウにセットする。

メニューのニューモニックスは、マウスを使用するのではなく、キーボードを使用してメニューをアクセスする方法を与える「Ａｌｔ．－，Ｆ．Ｎ」のようなキーシーケンスである。通常は、メニュー項目は、例えば、ウインドウズ３．１においてアンダーラインを付したその独特のニューモニックスとして指定された１つの文字を有する。メニュー項目をアクセスするために、その独特のニューモニックスがその収容するメニューのニューモニックスに添付される。全キーシーケンスの前にはシステムキーがあり、これはユーザがメニューニューモニックスをタイプ入力したい旨をシステムに知らせる。メニューニュモニックスは、イン

・プレース対話がその基礎となる幾つかのウインドウシステムで具現化されるときに特殊な問題を課する。より詳細には、変更されない条件のもとで、コンテニーオブジェクトがその位置でアクチベートされたときに、システムキー（ウインドウズ３．１における「ＡＬＴ」キー）及び他のナビゲーションキーを除く全てのキーボード入力がオブジェクトルートウインドウに対するウインドウ手順に送られる（図１３の項目１３０９を参照されたい）。というのは、オブジェクトルートウインドウには、オブジェクトがアクチベートされたときに入力フォーカスが与えられるからである。しかしながら、その基礎となるウインドウシステムはシステムキーをコンテナーアプリケーションフレームウインドウのウインドウ手順に送る。というのは、このキーは特殊に手渡しされるからである。従って、コンテナーアプリケーションは、システムキーシーケンスの選択されたメニュー項目に対応するキー入力を受け取ることはない。例えば、メニューニューモニックスのキーシーケンスがＡＬＴ．ー，ｍ（「ｍｏｖｅ」に対し）である場合には、「ＡＬＴ」キーの押圧がコンテナーアプリケーションへ送られそして「ー」及び「ｍ」キーの押圧がサーバアプリケーションへ送られる。この問題を解消するために、入力フォーカスはコンテナーアプリケーションフレームウインドウに一時的にセットされ、一方、フレームウインドウ手順はメニュー事象を処理する。この解決策の一例がライン５、３９及び４０に示されている。

ライン４ないし１４において、ハンドラーがメニューコマンドを受け取るときに、ハンドラーは、先ず、通常のプロセスを続けられるように入力フォーカスを回復させる。次いで、ハンドラーは、ＳａｖｅＭｅｎｕＲｏｕｔｉｎｅにセーブされた指示子をチェックする。これが、メニューがサーバアプリケーションに属することを指示する場合には、ハンドラーはオブジェクトウインドウのウインドウ手順を呼び出し、さもなくば、元のコンテナーアプリケーションウインドウ手順を呼び出す。より詳細には、ライン７において、ハンドラーは、コンテナーアプリケーションのフレームウインドウの特性として記憶されたオブジェクトのウインドウハンドルを得る。ライン８において、ハンドラーは、オブジェクトウイ

ンドウにメッセージを非同期で発信し、元のメッセージ及び元のパラメータを送

る。ライン１１において、指示子がサーバアプリケーションを特定しない場合には、ハンドラーは、コンテナーアプリケーションのフレームウインドウの元のウインドウ手順を検索し、そしてライン１２において、この手順を、指定されたメッセージ及びパラメータで呼び出す。コンテナーアプリケーションのフレームウインドウの元のウインドウ手順は、新たなウインドウ手順においてインストールされたＯｂｊｅｃｔＳｅｔＭｅｎｕＤｅｓｃｒｉｐｔｏｒへの呼び出しにおいてセーブされる。

ライン１５ないし２７において、ハンドラーは、他のメニュー関連メッセージを受け取ると、ライン６ないし１３で与えられたのと同様にコンテナー又はオブジェクトアプリケーションへメッセージを適宜発送する。ライン５０ないし５３において、他の種類のメッセージが受け取られた場合には、古いウインドウ手順が検索され、メッセージ及びパラメータがそこに送られる。

５．３　その位置でのデアクチベーション手順

ユーザが、その位置でアクチベートされるオブジェクト以外のエリアを選択すると、オブジェクトがデアクチベートされる。一般に、この振る舞いは、ユーザがコンテナーアプリケーションのウインドウか又はＭＤＩコンテナーアプリケーションの場合には異なる文書ウインドウの別のエリアにおいてマウスボタンでクリックするときに生じる。それ故、一般的に述べると、イン・プレース対話ＡＰＩのデアクチベーション方法は、マウスボタン事象を処理するためにコンテナーアプリケーションによって呼び出されるファンクションから呼び出される。

図３１は、Ｐｒｏｃｅｓｓ＿Ｍｏｕｓｅ＿ＬＢｕｔｔｏｎＤｏｗｎファンクションの具現化を示すフローチャートである。このファンクションは、左のボタンダウンメッセージの受信によって信号される入力事象を処理する。当業者であれば、このようなファンクションは、いずれかの種類の入力事象を受信した際に呼び出すことができそして他の事象は選択及び選択解除に使用できることが明らかであろう。ステップ１３０１では、このファンクションは、コンテナーアプリケーションがアクチベートしようとしていることを指示するフラグをセットする。このフラグは、ネスト状デアクチベーションの呼び出しのシーケンス中に使用さ

れ、ネスト状アクチベーションの場合に正しいユーザインターフェイスリソースが表示されるように確保する。ステップ３１０２において、このファンクションは、フラグＡＣＴＩＶＡＴＥＤ＿ＳＯＭＥＯＮＥをチェックして、オブジェクトがその位置でアクチベートされたかどうかを判断し、もしそうであれば、ステップ３１０４へ続き、さもなくば、ステップ３１０３へ続く。ステップ３１０３では、このファンクションは、その通常の左ボタンダウン事象処理を実行し、そして復帰となる。ステップ３１０４では、このファンクションは、現在アクチベートされたオブジェクトに対しＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔインターフェイスを検索する。ステップ３１０５において、このファンクションは、オブジェクトのＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：ＩｎＰｌａｃｅＵＩＤｅａｃｔｉｖａｔｅ方法を呼び出す。このファンクションは、次いで、ステップ３１０３へ続く。

図２１に示すように、現在アクチベートされたオブジェクトのＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：ＩｎＰｌａｃｅＵＩＤｅａｃｔｉｖａｔｅ方法は、次いで、そのＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ：：ＯｎＵＩＤｅａｃｔｉｖａｔｅ方法を呼び出して、コンテナーアプリケーションがそのユーザインターフェイスリソースをインストールできるようにする。このデアクチベーションは、最上位レベルのコンテナーはアクチベートしようとしているオブジェクトのコンテナーのいずれかに到達するまで上方にネスト構成になる。（図１９を参照されたい）。例えば、図４を参照すると、埋め込まれたスプレッドシートオブジェクト４０５内に示された埋め込まれたチャートオブジェクト４０９が現在アクチベートされたオブジェクトでありそしてユーザがスプレッドシートオブジェクト４０５を選択してこれをその位置でアクチベートする場合は、チャートのＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：ＩｎＰｌａｃｅＵＩＤｅａｃｔｉｖａｔｅ方法が呼び出され、これは、次いでスプレッドシートのＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：ＯｎＵＩＤｅａｃｔｉｖａｔｅ方法を呼び出す。この後者の方法は図４に示すようにスプレッドシートオブジェクトに対するユーザインターフェイスをインストールする。一方、ユーザが複合文書内のどこかでクリックした場合には、スプレッドシートのＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：ＯｎＵＩＤ

ｅａｃｔｉｖａｔｅ方法は、複合文書のＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：ＯｎＵＩＤｅａｃｔｉｖａｔｅ方法を呼び出す。この後者の方法は、ワードプロセスユーザインターフェイスをインストールする。というのは、このオブジェクトは最上位レベルのコンテナーオブジェクトだからである。

コンテナーアプリケーションは、表示されたコンテナーオブジェクトを垂直又は水平にスクロールするためのスクロールバーを表示することができる。これらスクロールバーは、コンテナーウインドウの一部分である。コンテナーオブジェクトがアクチベートされたコンテニーオブジェクトを有する場合には、ユーザがコンテニーオブジェクトの外側のエリアでクリックしたときに、コンテニーオブジェクトはデアクチベートされる。好ましい実施例では、コンテナーアプリケーションは、スクロールバーメッセージを受け取る際に、コンテニーオブジェクトをデアクチベートしない。むしろ、コンテナーアプリケーションは、スクロールを行い、入力フォーカスがコンテニーオブジェクトと共に留まるよう確保する。

５．４　コンテナーアプリケーションを閉じる

コンテナーアプリケーションが、既にアクチベートされたオブジェクトを再アクチベートする開放（undo）オペレーションをもはや実行できなくなった後であって、且つコンテナーアプリケーションがユーザによって閉じられる前のある時間に、コンテナーアプリケーションは、既にアクチベートされたオブジェクトに関連したユーザインターフェイスリソースを永久的に割り当て解除する。これらのリソースを割り当て解除するために、コンテナーアプリケーションは、既にアクチベートされたオブジェクトに関連したＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：ＩｎＰｌａｃｅＤｅａｃｔｉｖａｔｅ方法を呼び出す。この方法は、次いで、共用メニューデータ構造と、複合メニューバーに関連したメニューとを割り当て解除する。（図２０及びそれに関連した説明を参照されたい。）

５．５　モードレスダイアログとの対話

その位置でアクチベートされたオブジェクトであってそのサーバアプリケーションがモードレスダイアログを表示しているようなオブジェクトとユーザが対話するときに、ユーザがそれ自身のモードダイアログを表すコンテナーアプリケーションメニューからメニュー項目を選択したい場合には、コンテナーアプリケー

ションは、サーバアプリケーションのモードレスダイアログを一時的に隠す。モードレスダイアログボックスが隠されるのは、ユーザが同時に表示される２つのダイアログボックスを見て混乱しそしてどのボックスに入力が送られるか分からないからである。というのは、サーバ及びコンテナーアプリケーションは、１つのアプリケーションとして現れることを意味するからである。又、モードダイアログは、他のダイアログとの競合を回避するようにプログラムされていない。というのは、慣例的なアプリケーションにおいては、その基礎となるウインドウシステムが単一のアプリケーション内のモードダイアログの外側の入力を禁止するからである。イン・プレース対話を使用する場合には、２つのアプリケーションが共働して１つに見えるので、このような競合は自動的に回避される。それ故、アプリケーションは、モードダイアログとモードレスダイアログとの間の競合を回避するように共働しなければならない。例えば、ユーザがスプレッドシートアプリケーションのＥｄｉｔメニューにおける「Ｆｉｎｄ．．．」メニュー項目を選択し、サーバアプリケーションによってモードレスダイアログが表示されると仮定する。ここで、ユーザが複合文書の一部分をプリントアウトしようとし、従って、ユーザが、ワードプロセス（コンテナー）アプリケーションに属するＦｉｌｅメニュー上の「Ｐｒｉｎｔ．．．」メニュー項目を選択すると仮定する。ワードプロセスアプリケーションは、両方のダイアログが同時に表示されないのが好ましいので、「Ｆｉｎｄ．．．」ダイアログを隠す。これを行うために、ワードプロセスアプリケーションは、スプレッドシートアプリケーションのＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔ：：ＥｎａｂｌｅＭｏｄｅｌｅｓｓ方法を呼び出して、モードレスダイアログを隠すように要求する。次いで、コンテナーアプリケーションが「Ｐｒｉｎｔ．．．」ダイアログの処理を終了した後に、ＥｎａｂｌｅＭｏｄｅｌｅｓｓ方法を呼び出して、モードレスダイアログを再表示する。

サーバアプリケーションがコンテナーアプリケーションのモードレスダイアログを隠す必要がある場合にも同様の状態が生じ得る。この場合には、ＩＯＬＥＩｎＰｌａｃｅＦｒａｍｅ：：ＥｎａｂｌｅＭｏｄｅｌｅｓｓ方法が使用される。

５．６　アクセラレータキーの組合せの取り扱い

本発明の好ましい実施例においては、その基礎となるウインドウシステムは、

アクセラレータキー組合せと称する概念をサポートし、ユーザがキーボードショーとカットを介してメニューコマンドを呼び出すことができるようにする。アクセラレータキー組合せは、特定のメニューコマンドを呼び出すことと同等であるようにアプリケーションによって指定された一連のキーである。例えば、「ＣＮＴＲＬ」キーを押した後に「N」キーを押すことより成るキーのシーケンスは、「Ｆｉｌｅ」メニュー上のメニューコマンド「Ｎｅｗ」に変換される。典型的なシステムにおいて、アクセラレータキー組合せは、ユーザによって指定でき、アプリケーション内で独特である必要がある。

一般に、アクセラレータキー組合せ（アクセラレータ）は、アプリケーションのメッセージポンプにおいて処理される（図１４のステップ１４０２を参照）。典型的なメッセージポンプは、その基礎となるウインドウシステムを呼び出してアクセラレータ変換テーブルをそこに通し、ウインドウシステムが、アクセラレータがどのメニュー項目コマンドに対応するかを決定するようにさせる。ウインドウシステムは、次いで、それにより得たメニューコマンドを適当なウインドウ手順に送信する。

オブジェクトリンク及び埋め込みイン・プレース対話ＡＰＩではアクセラレータに対して問題が生じ得る。先ず、サーバアプリケーションはそのコンテナーアプリケーションのプロセススペース内でオブジェクトハンドラーとして具現化できるので、コンテナーアプリケーションは、サーバアプリケーションがそれ自身のアクセラレータを変換する機会をもつよう確保しなければならない。好ましくは、サーバアプリケーションには、サーバアプリケーションがその位置でアクチベートされたときにあいまいなアプリケーションアクセラレータの処理においてコンテナーアプリケーションより高い優先順位が与えられる。又、サーバアプリケーションがそれ自身のプロセススペースにおいて具現化される場合には、それが確認できないアクセラレータをコンテナーアプリケーションに通さねばならない。

この問題を解決するために、コンテナーアプリケーション及びサーバアプリケーションのメッセージポンプは、アプリケーションアクセラレータを変換する機

会を互いに許すように変更される。コードテーブル６及び７は、サーバアプリケーションが個別のプロセスとして実行されるときに適用できるサーバアプリケーションのメッセージポンプに対する変更を示している。更に、コードテーブル８は、サーバアプリケーションが同じプロセス内でコンテナーアプリケーションとして（オブジェクトハンドラーとして）実行されるときに適用できるコンテナーアプリケーションのメッセージポンプに対する変更を示している。

Code Table 6

```
Object's message loop:
...
1   while (GetMessage (&msg. NULL. NULL. NULL) {
2       if (not (TranslateAccelerator (hwndObj. hAccel. &msg)) {
3           if (not (ObjectTranslateAccelerator (&msg. hwndFrame. hAccel Table)) {
4               TranslateMessage (&msg);
5               DispatchMessage (&msg);
6               }}}
```

コードテーブル６は、その位置でアクチベートされたオブジェクトのメッセージポンプに対する変更の具現化を示す。これらの変更は、サーバアプリケーションが、コンテナーアプリケーションに、サーバアプリケーション（個別のプロセス）が到来メッセージを最終的に破棄する前にアプリケーションアクセラレータを変換する機会を与えることができるようにする。ライン２において、サーバアプリケーションは、それ自身の変換テーブル（変数ｈＡｃｃｅｌに記憶された）を用いてアクセラレータを変換するよう試みる。ライン３において、変換ずべきアクセラータがなかったか、又はサーバアプリケーションの変換テーブルにアクセラレータが見つからなかったことにより、この変換が首尾良くいかなかった場合には、サーバアプリケーションは、特殊なオブジェクトリンク及び埋め込みＡＰＩのフアンクションＯｂｊｅｃｔＴｒａｎｓｌａｔｅＡｃｃｃｌｅｒａｔｏ

ｒを呼び出す。このファンクションＯｂｊｅｃｔＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒは、コンテナーアプリケーションによってアクセラータが所望されるかどうか決定し、もしそうならば、遠隔手順通話を介してコンテナーアプリケーションにメッセージを送信し、アクセラレータを変換するように要求する。遠隔手順通話メカニズムは、その同期特性により、発呼者（サーバプロセス）が

更に入力を受け取る前にコンテナーアプリケーションがメッセージを処理して復帰するよう確保する。ライン４ないし５において、コンテナーアプリケーションがアクセラレータを変換しなかった場合には、サーバアプリケーションが入力メッセージをその通常の形態で処理する（それをフィルタして発送する）。

Code Table 7

```
ObjectTranslateAccelerator(lpmsg. hwndFrame. hAccelFrame){
1   if (keystroke in lpmsg not found in hAccelFrame)
2       return (false)
3   else {
4       Send RPC message to hwndFrame to invoke
5           hwndFrame -> TranslateAccelerator
6       return (value from RPC call)
    }}
```

コードテーブル７は、オブジェクトリンク及び埋め込みＡＰＩのファンクションＯｂｊｅｃｔＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒの具現化を示している。このファンクションは、サーバアプリケーションがコンテナーアプリケーションにアクセラレータを処理する機会を与えることができるようにする。同期メッセージ処理に本来ある落とし穴（不定の待機のような）を回避するために、ＯｂｊｃｃｔＴｒａｎｓｌａｔｃＡｃｃｅｌｅｒａｔｏｒは、先ず、コンテナーアプリケーションのＴｒａｎｓｌａｔｅＡｃｃｅｌｃｒａｔｏｒ方法を呼び出そうと試みる前にコンテナーアプリケーションがアクセラレータに挿入されたかどうか調べるチェックを行う。コンテナーアプリケーションのアクセラレータテーブルは、サーバアプリケーションによって通された指定のパラメータである。これは、ＩＯＬＥＩｎＰｌａｃｅＰａｒｅｎｔ：：ＧｅｔＷｉｎｄｏｗＣｏｎｔｅｘｔに対する通話を介してサーバアプリケーションにアクセスできる。コンテナーアプリケーションのＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒ方法が呼び出された場合には、このファンクションは、コンテナーアプリケーションにより返送された値をサーバアプリケーションへ戻し、従って、サーバアプリケーションはメッセージを適切に破棄することができる。

Code Table 8

```
Container's message loop:
...
1   while (GetMessage (&msg. NULL. NULL. NULL) {
2       pwhObj = GetActiveObjectHwnd (hwndDoc):
3       translated = false:
4       if ((pwhObject ! = NULL) {
5               pipObj = determine the IOLEInPlaceObject interface for pwhObj:
6               if (pipObj -> TranslateAccelerator (&msg))
7                   translated = true;
8               }
9       if (not (translated)) {
10          if ( not (TranslateMDISysAccel (hwndMDIcontainer. &msg))
11              if (not (TranslateAccelerator (hwndFrame. hAccel. &msg)) {
12                  TranslateMessage (&msg):
13                  DispatchMessage (&msg):
                }}}}
```

コードテーブル８は、イン・プレース対話をサポートするコンテナーアプリケーションのメッセージポンプの典型的な具現化を示す。これらの変更は、コンテナーアプリケーションが、サーバアプリケーション（コンテナーアプリケーションと同じプロセス内で実行している）に、コンテナーアプリケーションが到来メッセージを完全に破棄ずる前にアプリケーションアクセラレータを変換する機会を与えることができるようにする。ライン２において、コードは、コンテナーアプリケーションの文書ウインドウに関連した現在アクティブなオブジェクトウインドウハンドルを検索する。ライン４ないし８において、アクティブなオブジェクトウインドウハンドルがない場合には、コードは、そのオブジェクトウインドウハンドルに対応するＩＯＬＥＩｎＰｌａｃｅＯｂｊｅｃｔＴｒａｎｓｌａｔｅＡｃｃｅｌｅｒａｔｏｒ方法を呼び出して、サーバアプリケーションがアクセラレータキー組合せを変換できるようにする。ライン９ないし１１において、サーバアプリケーションがアクセラレータを変換しないか、又はアクティブなオブジェクトがない場合には、コンテナーアプリケーションは、それ自身の変換テーブル（変数ｈＡｃｃｅｌに記憶された）を用いてアクセラレータを変換するよう試みる。ライン１２ないし１３において、変換すべきアクセラレータが確認されない場合には、コンテナーアプリケーションは、入力メッセージを通常の形態で処理する（それをフィルタして発送する）。

以上、好ましい実施例について本発明を説明したが、本発明は、この実施例に限定されるものではない。本発明の精神内での変更が当業者に明らかであろう。本発明の範囲は、以下の請求の範囲によって限定されるものとする。

【図1】

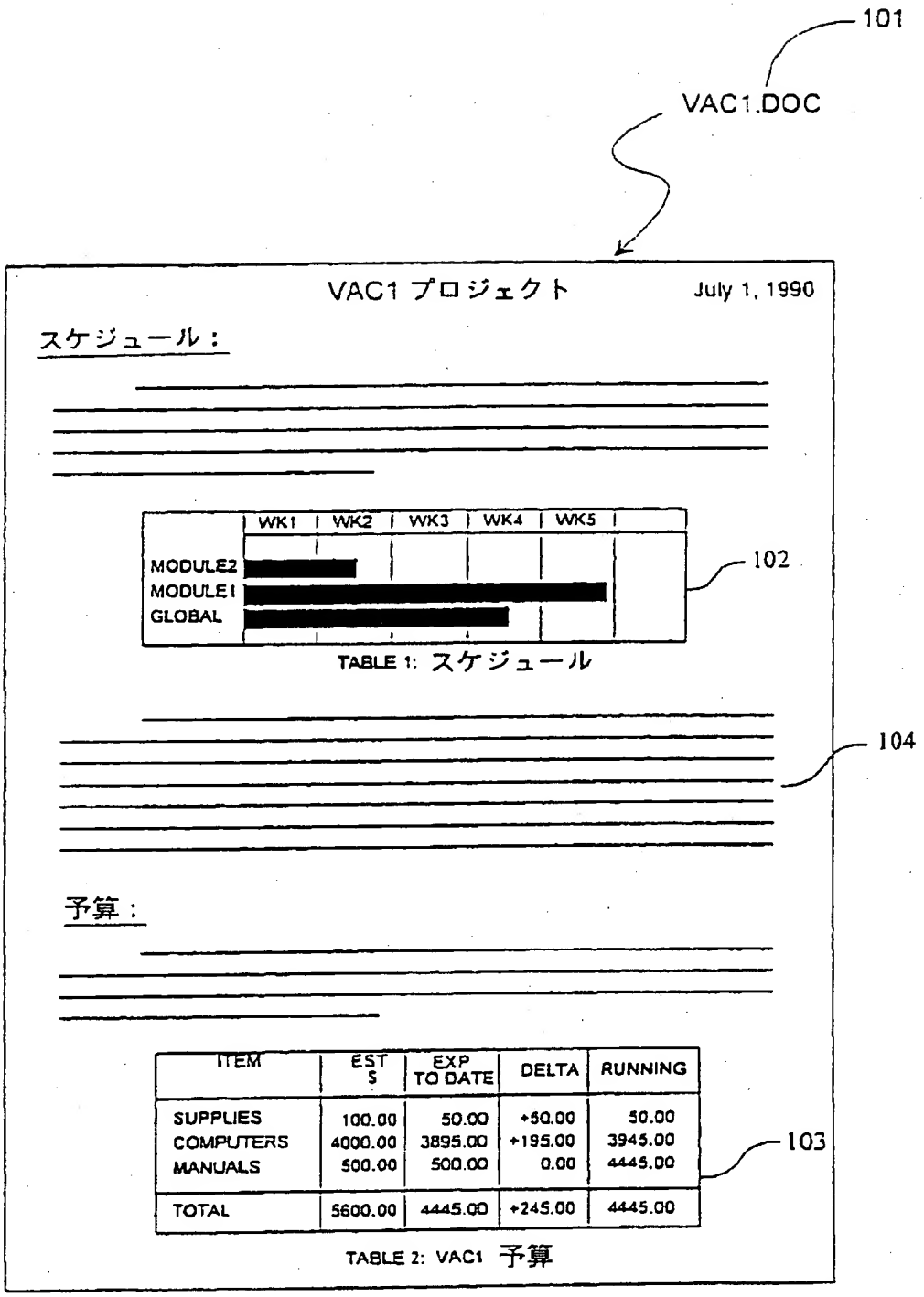
101
VAC1.DOC
VAC1 プロジェクト
July 1, 1990
スケジュール：
WK1
WK2
WK3
WK4
WK5
MODULE2
MODULE1
GLOBAL
102
TABLE 1: スケジュール
104
予算：
ITEM
EST $
EXP TO DATE
DELTA
RUNNING
SUPPLIES 100.00 50.00 +50.00 50.00
COMPUTERS 4000.00 3895.00 +195.00 3945.00
MANUALS 500.00 500.00 0.00 4445.00
TOTAL 5600.00 4445.00 +245.00 4445.00
103
TABLE 2: VAC1 予算

Figure 1

【図２】

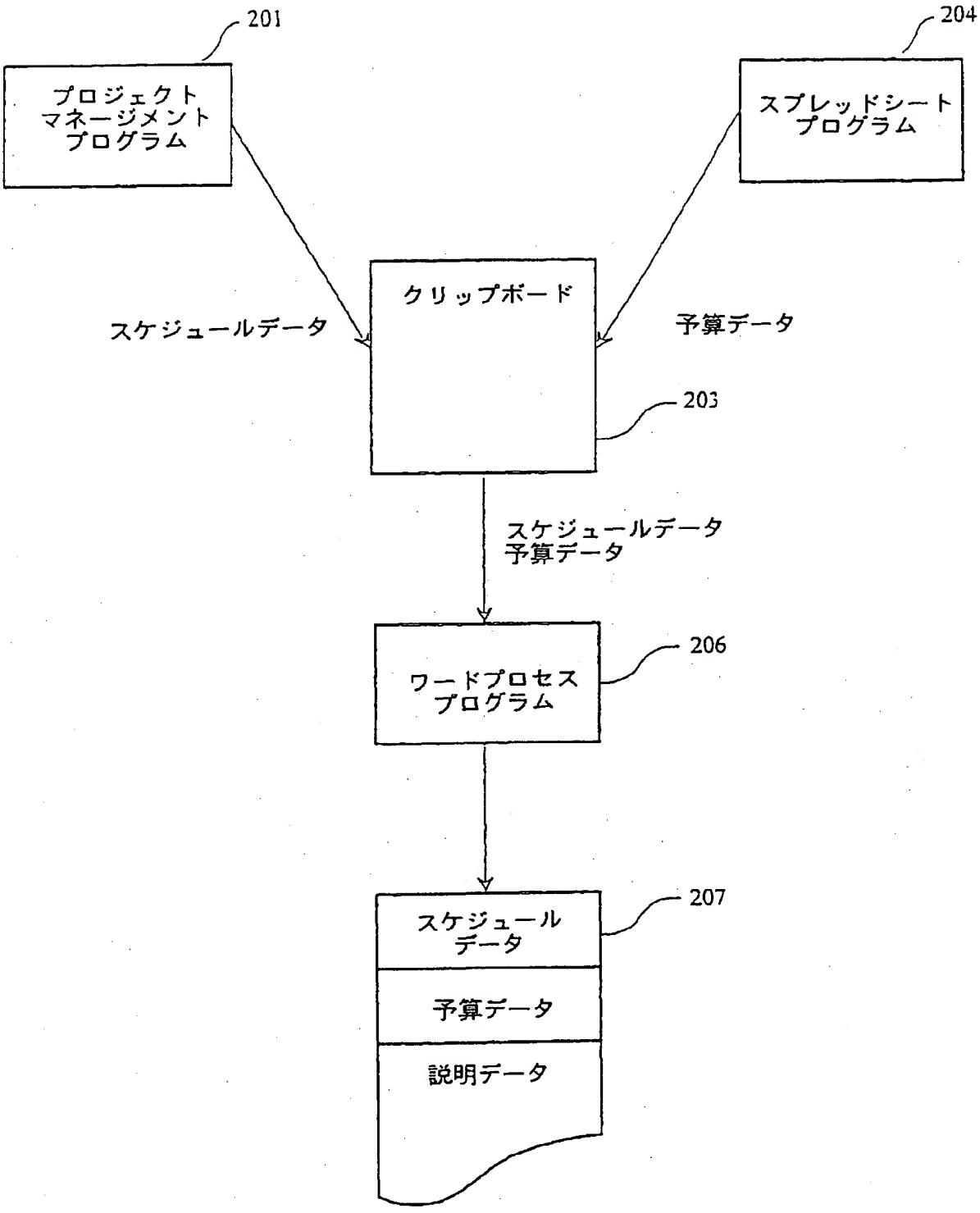
201
プロジェクト
マネージメント
プログラム
204
スプレッドシート
プログラム
クリップボード
スケジュールデータ
予算データ
203
スケジュールデータ
予算データ
206
ワードプロセス
プログラム
207
スケジュール
データ
予算データ
説明データ

Figure 2

【図3】

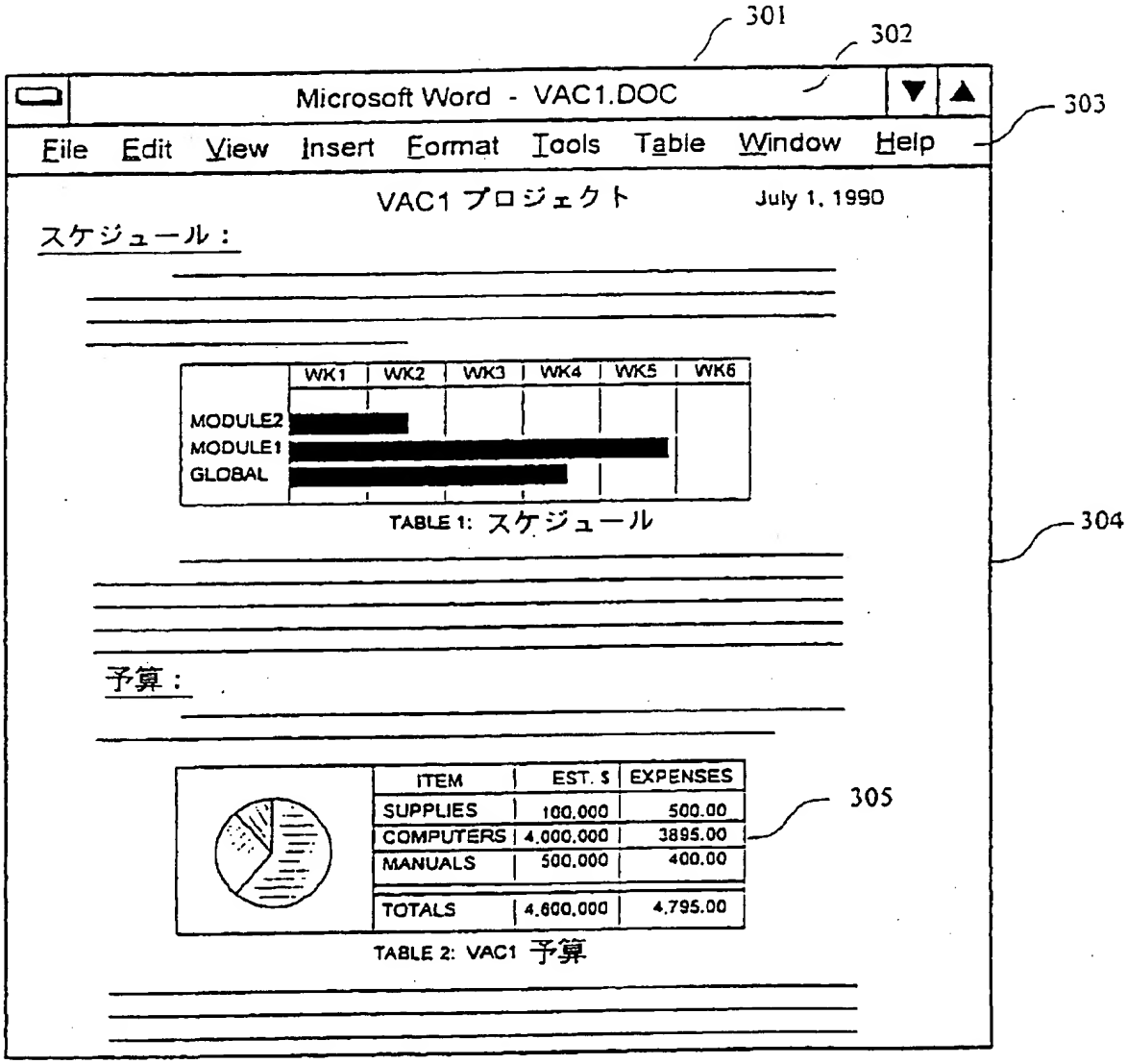
301
302
Microsoft Word - VAC1.DOC
303
File Edit View Insert Format Tools Table Window Help
VAC1 プロジェクト
July 1, 1990
スケジュール：
WK1 WK2 WK3 WK4 WK5 WK6
MODULE2
MODULE1
GLOBAL
TABLE 1: スケジュール
304
予算：
ITEM EST. $ EXPENSES
SUPPLIES 100,000 500.00
COMPUTERS 4,000,000 3895.00
MANUALS 500,000 400.00
TOTALS 4,600,000 4,795.00
305
TABLE 2: VAC1 予算

Figure 3

【図４】

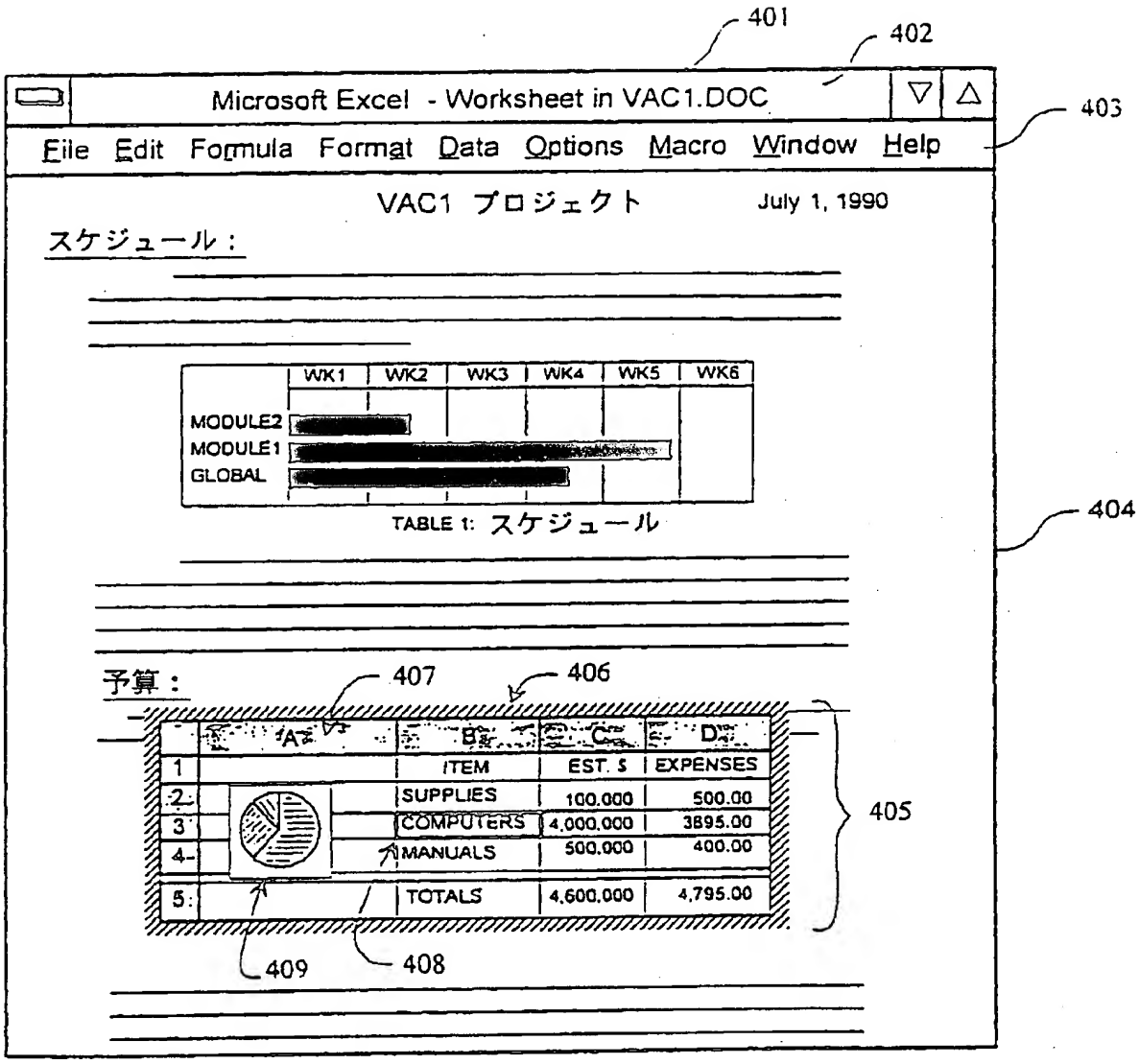
401
402
Microsoft Excel - Worksheet in VAC1.DOC
403
File Edit Formula Format Data Options Macro Window Help
VAC1 プロジェクト
July 1, 1990
スケジュール：
WK1 WK2 WK3 WK4 WK5 WK6
MODULE2
MODULE1
GLOBAL
TABLE 1: スケジュール
404
予算：
407
406
A B C D
1 ITEM EST. $ EXPENSES
2 SUPPLIES 100.000 500.00
3 COMPUTERS 4,000.000 3895.00
4 MANUALS 500.000 400.00
5 TOTALS 4,600.000 4,795.00
405
409
408

Figure 4

【図５】

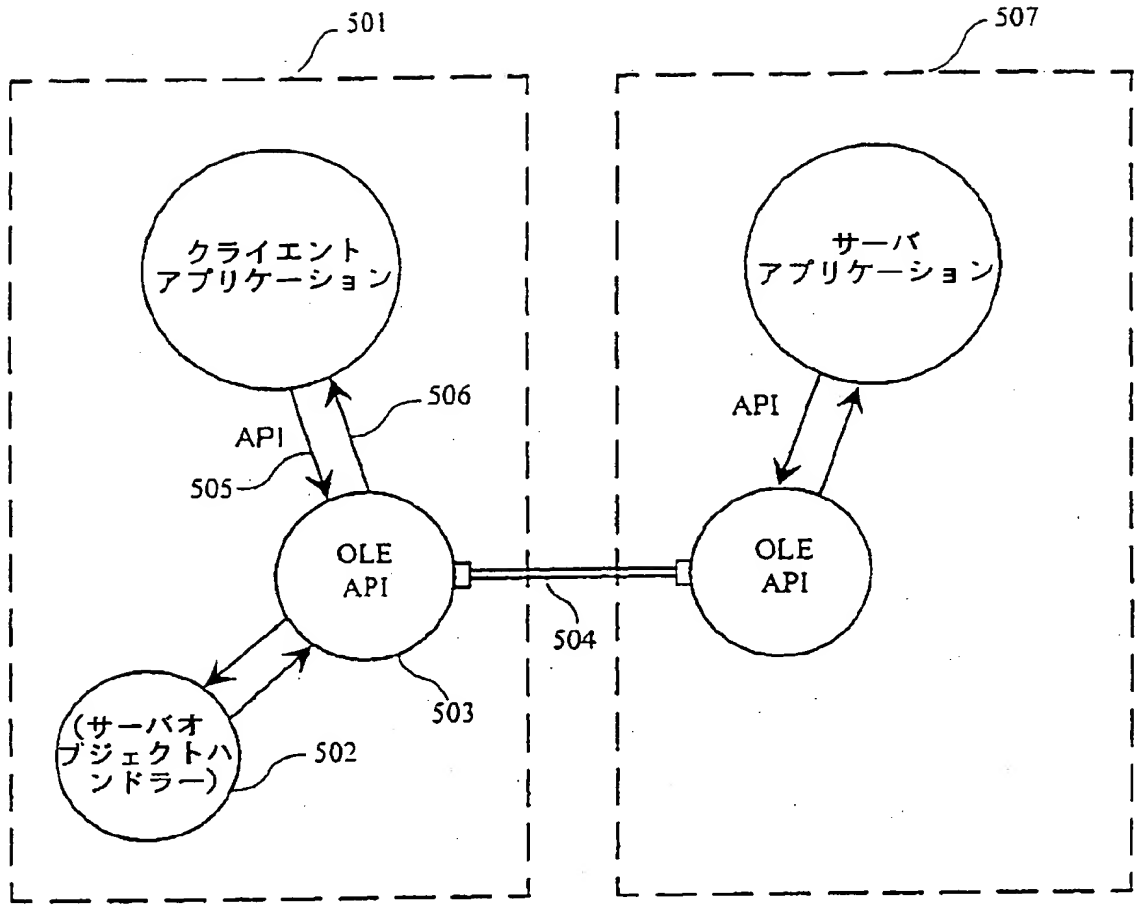
501
507
クライエント
アプリケーション
サーバ
アプリケーション
506
API
API
505
OLE
API
OLE
API
504
503
(サーバオ
ブジェクトハ
ンドラー)
502

Figure 5

【図６】

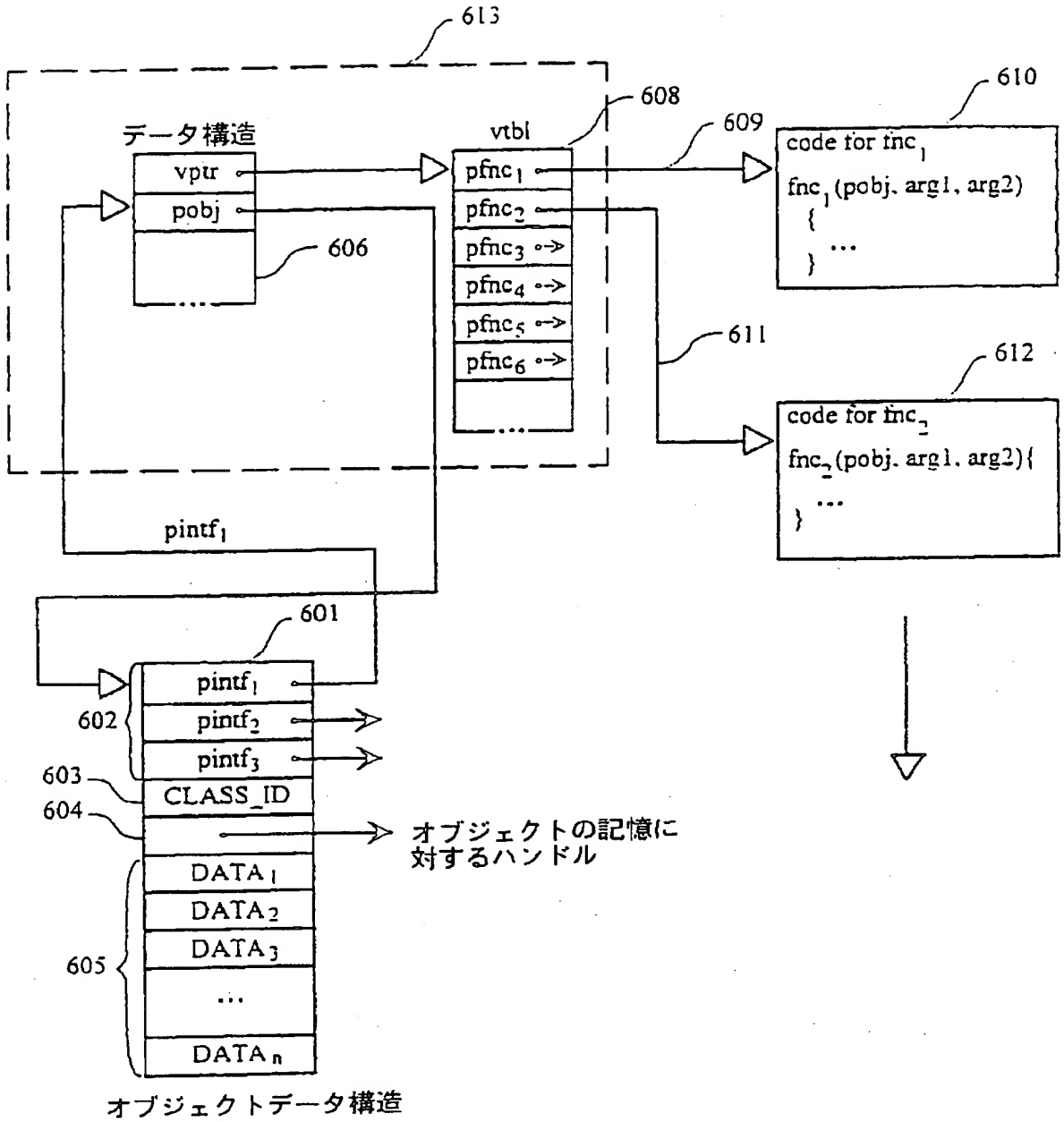
613
608
610
609
データ構造
vtbl
vptr
pobj
606
pfnc1
pfnc2
pfnc3
pfnc4
pfnc5
pfnc6
code for fnc1
fnc1(pobj, arg1, arg2)
{
...
}
611
612
code for fnc2
fnc2(pobj, arg1, arg2){
...
}
pintf1
601
pintf1
602
pintf2
pintf3
603
CLASS_ID
604
オブジェクトの記憶に
対するハンドル
DATA1
DATA2
DATA3
605
...
DATAn
オブジェクトデータ構造

Figure 6

【図７】

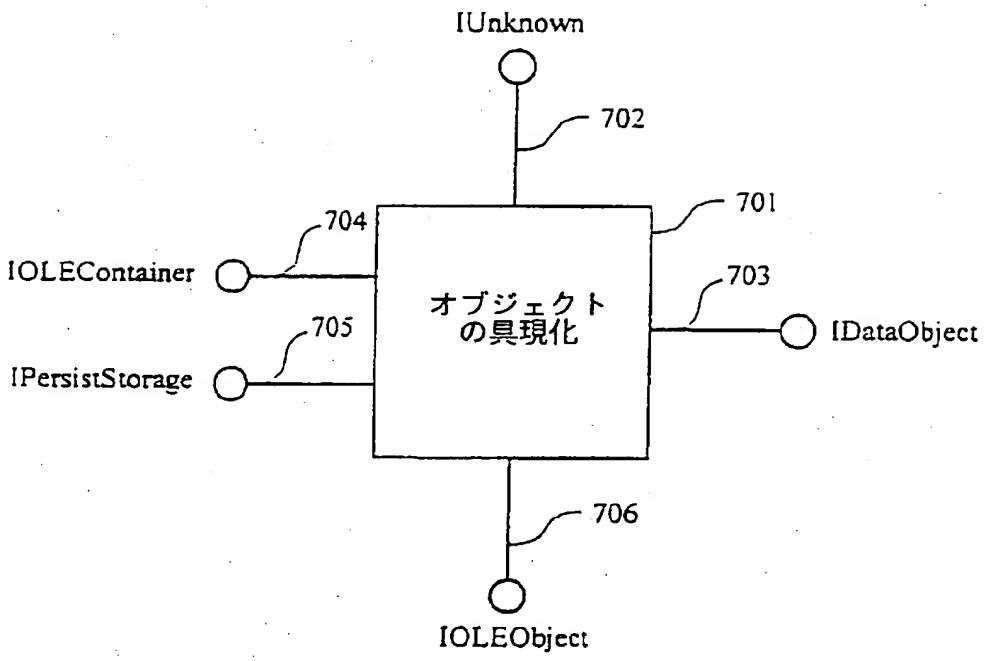

Figure 7

【図８】

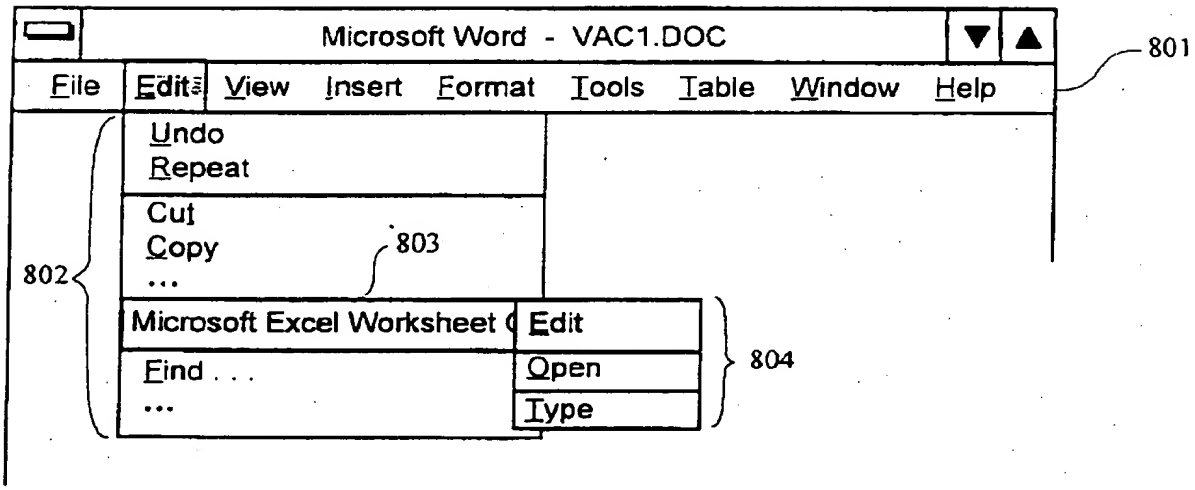

Figure 8

【図９】

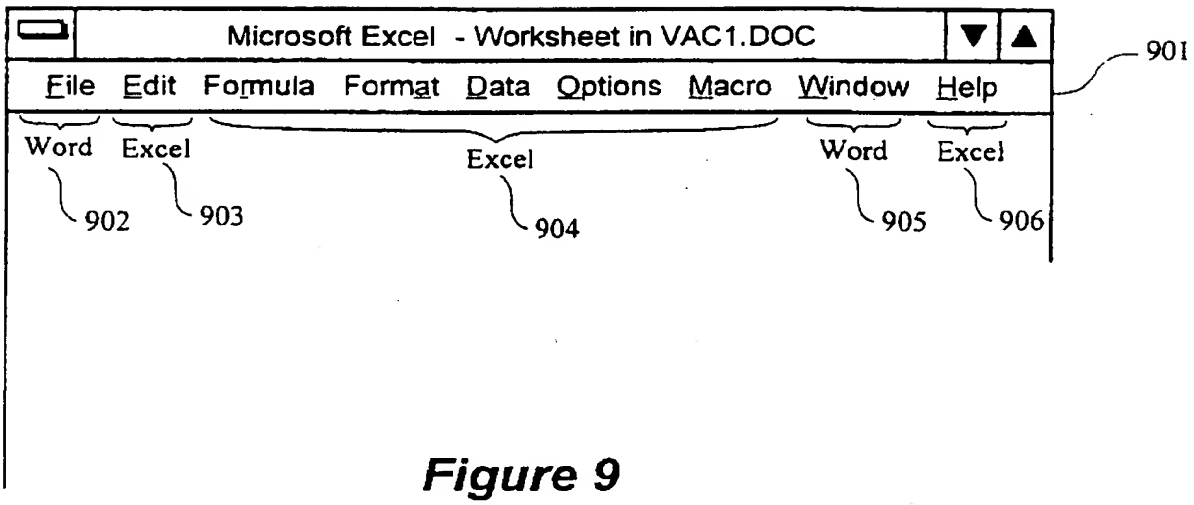
Microsoft Excel - Worksheet in VAC1.DOC
File Edit Formula Format Data Options Macro Window Help
901
Word
Excel
Excel
Word
Excel
902
903
904
905
906

Figure 9

【図１０】

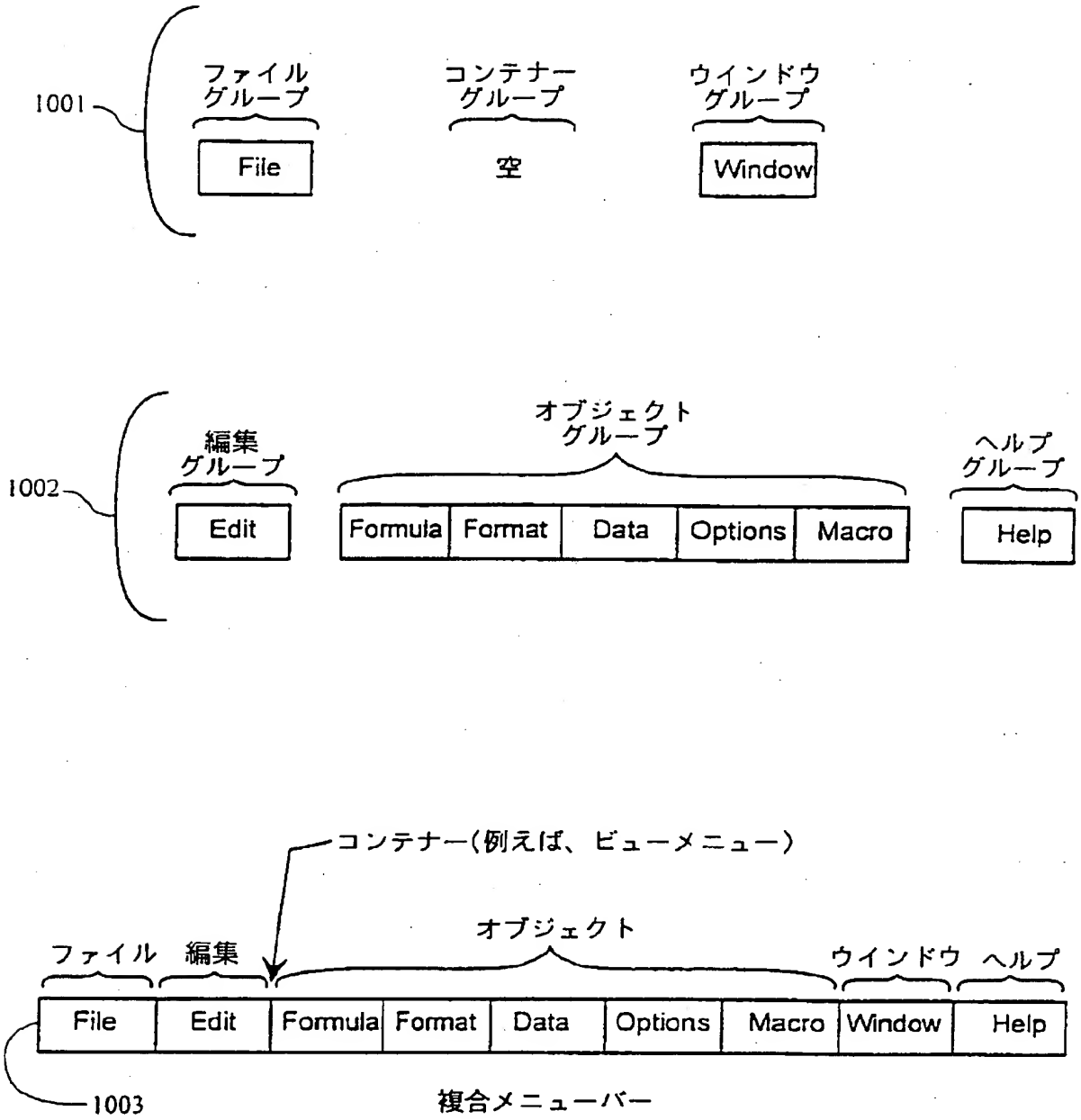

Figure 10

【図１１】

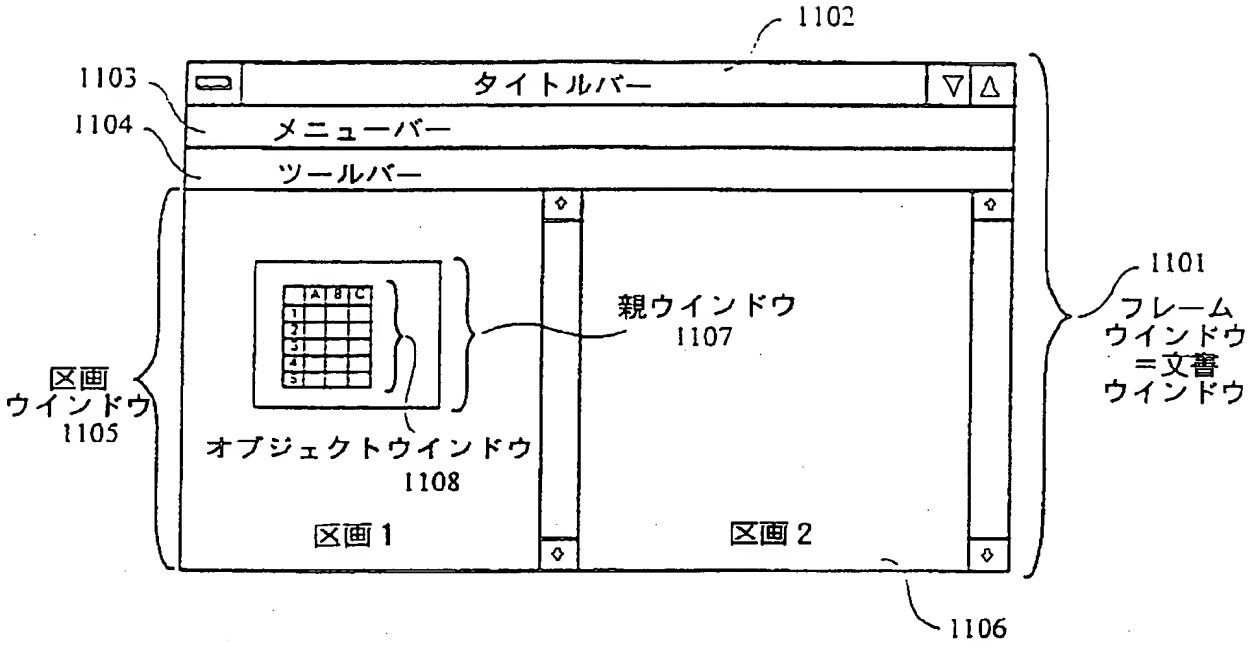

Figure 11

【図１２】

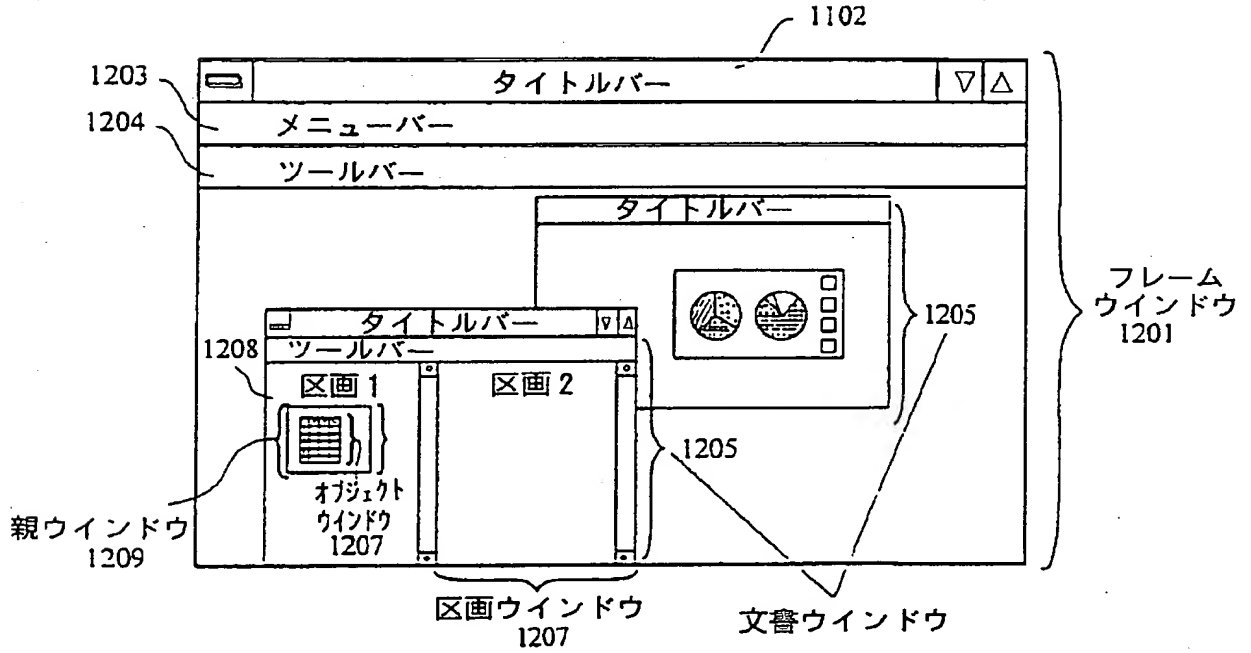

Figure 12

【図１３】

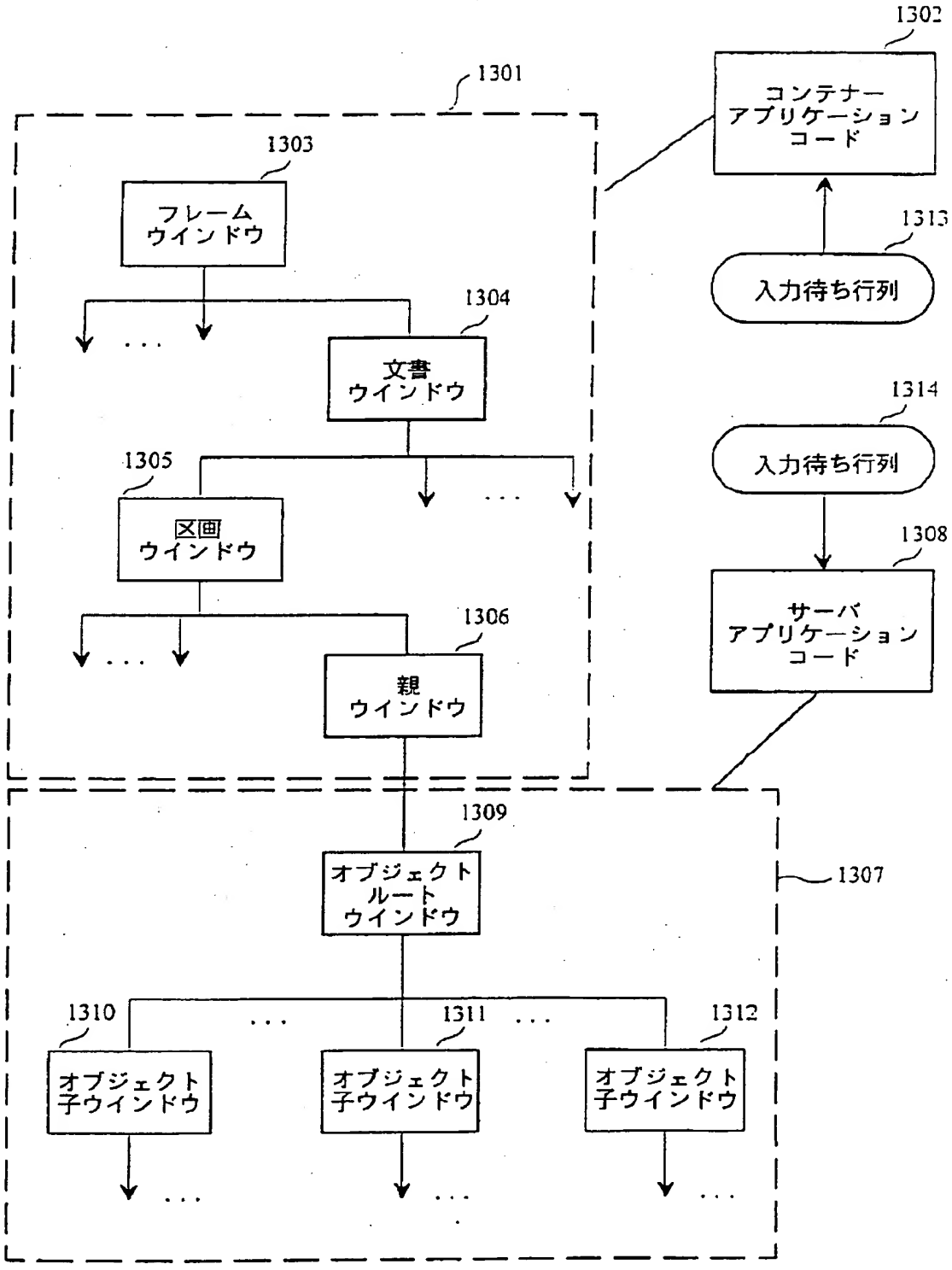

Figure 13

【図１４】

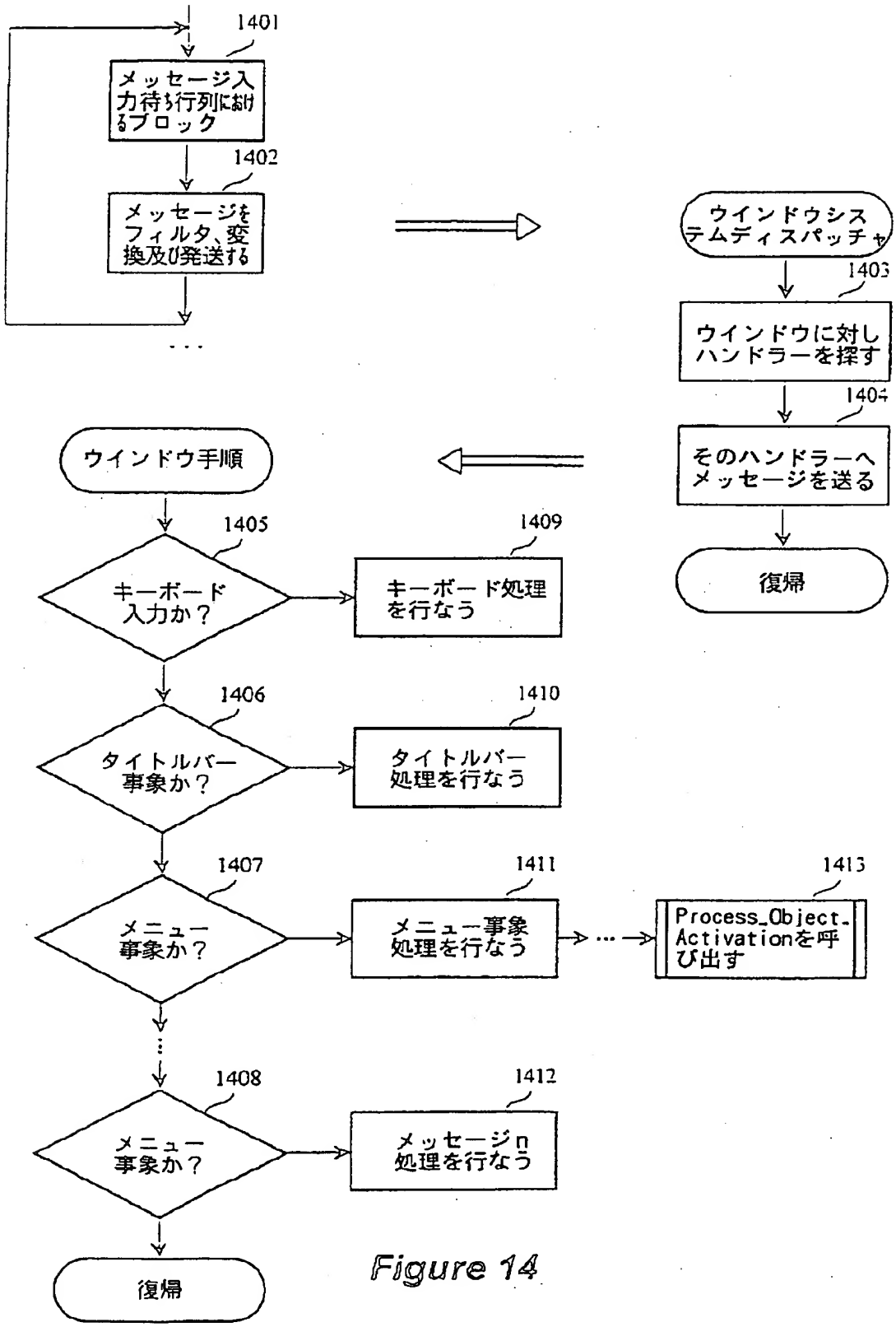

Figure 14

【図１４Ｂ】

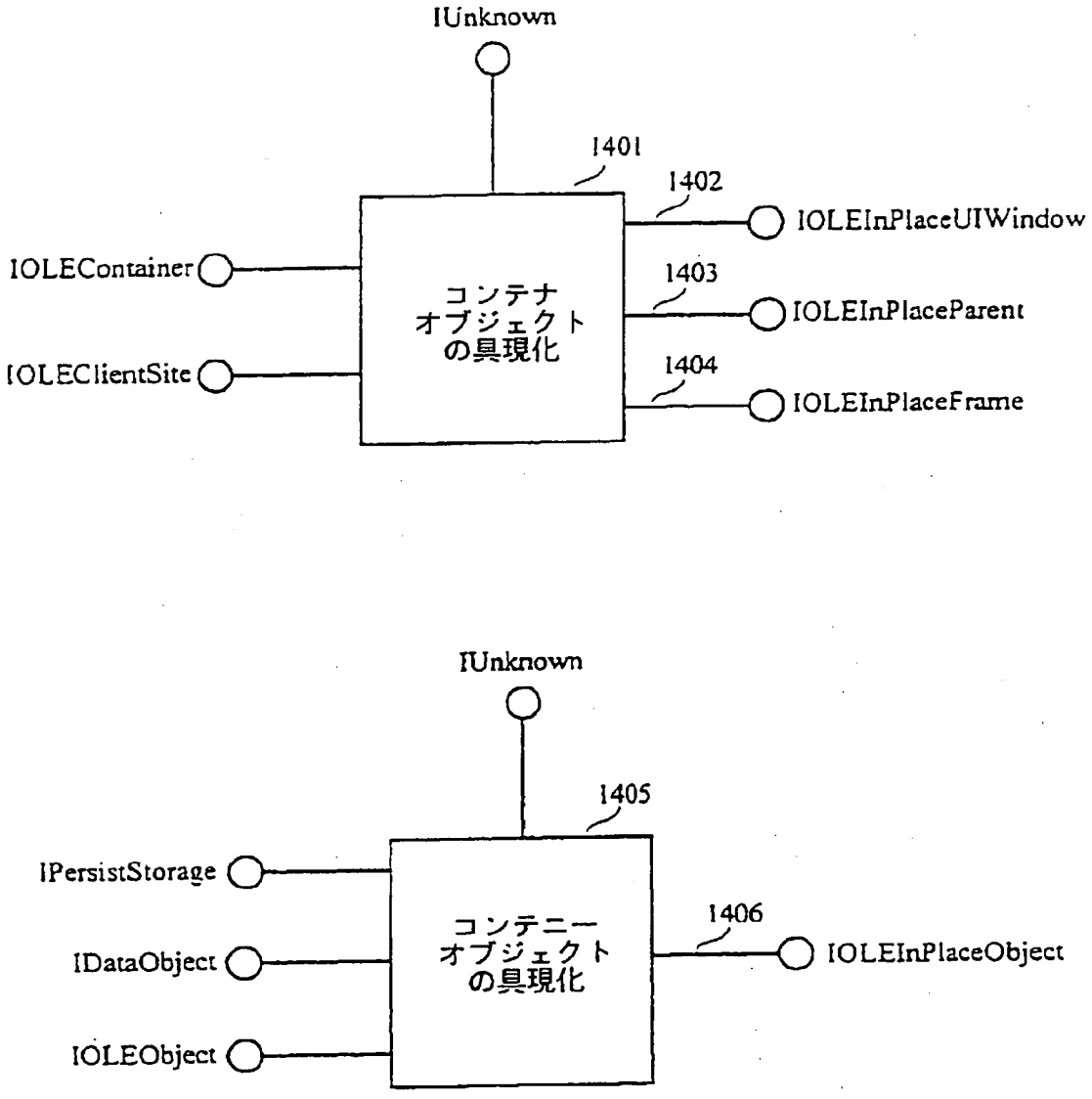

Figure 14B

【図１５】

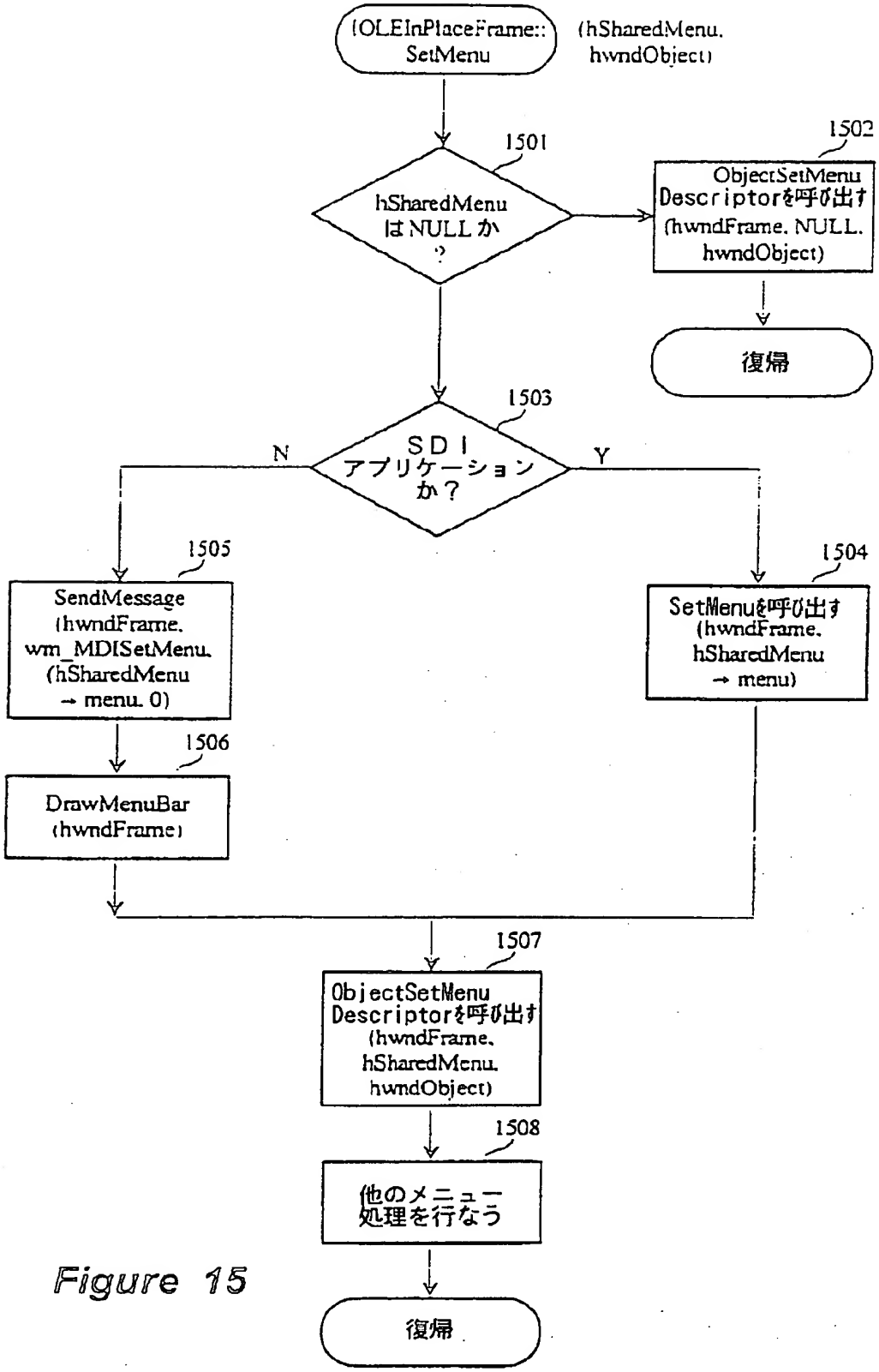

Figure 15

【図１６】

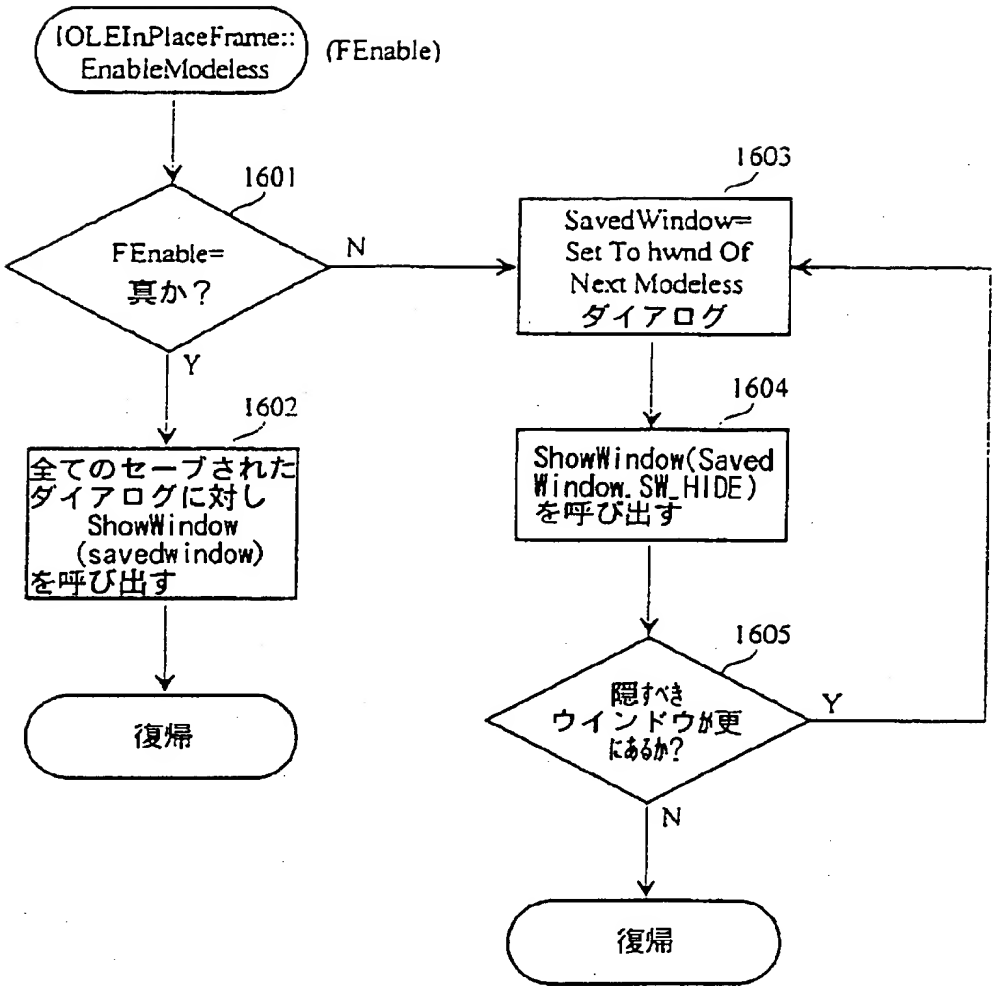

Figure 16

【図１７】

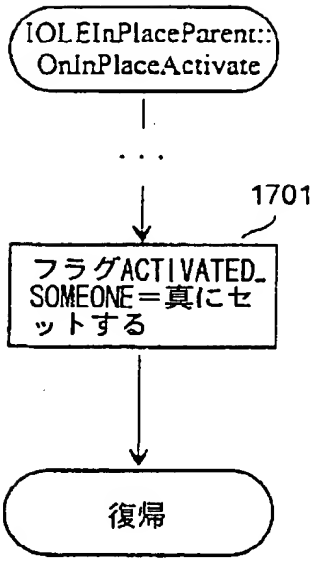

Figure 17

【図１８】

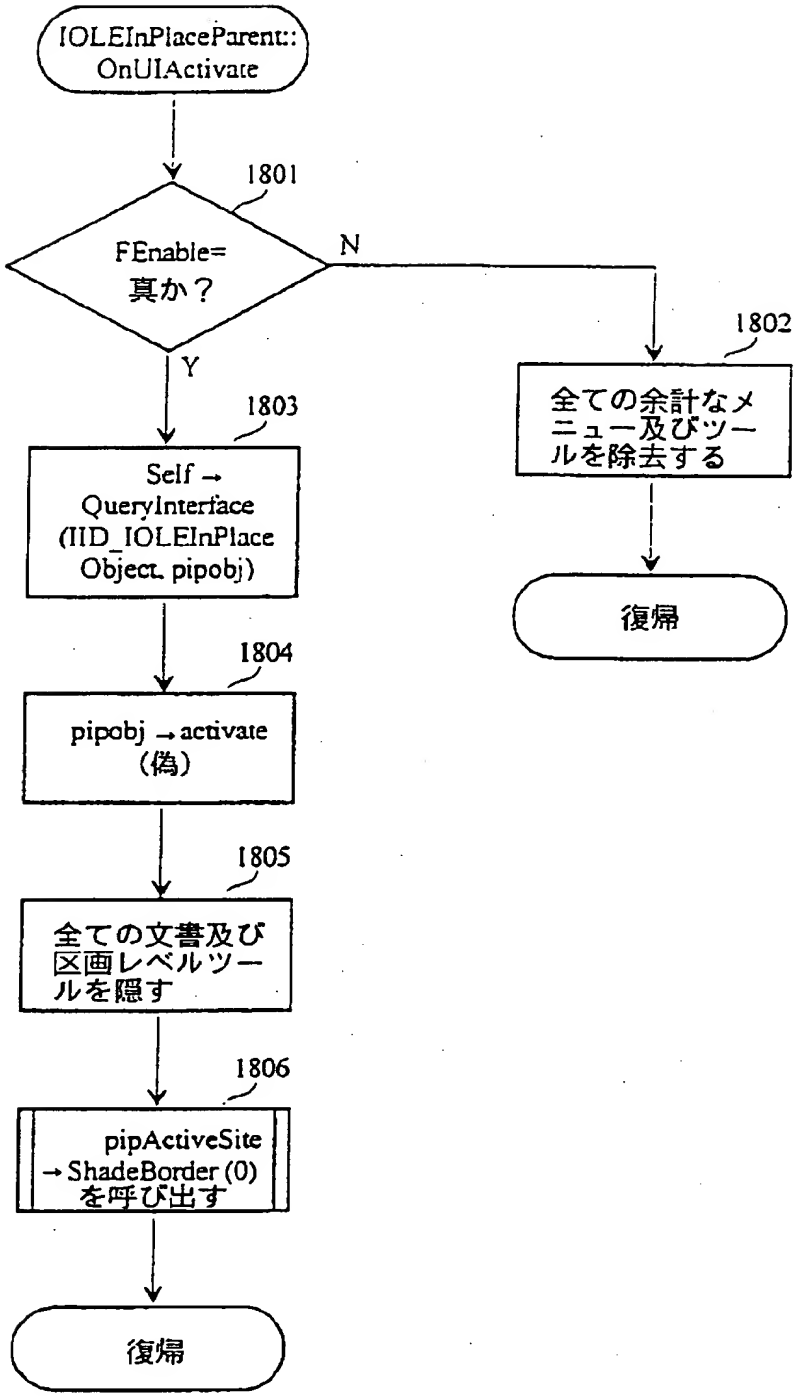

Figure 18

【図１９】

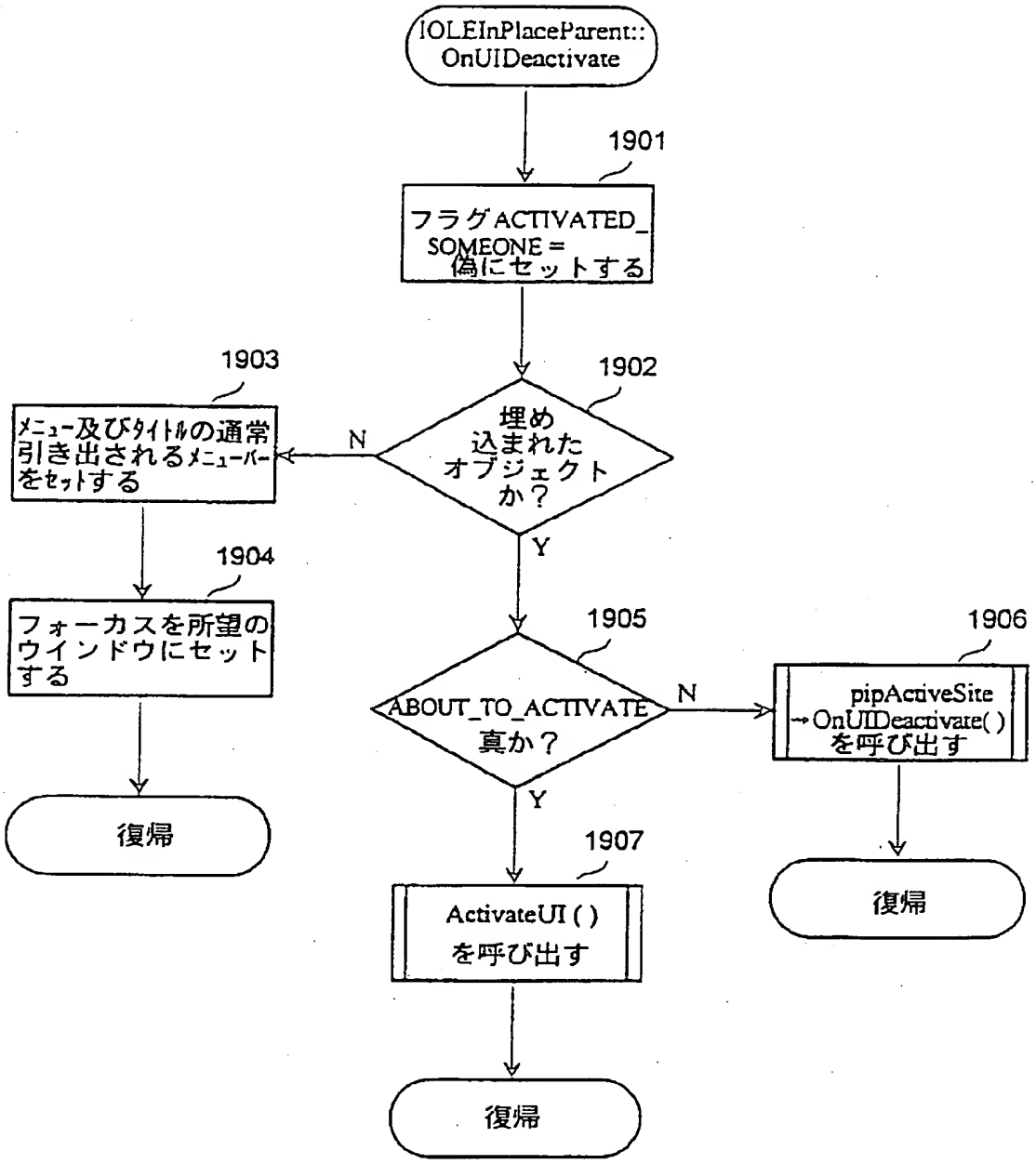

Figure 19

【図２０】

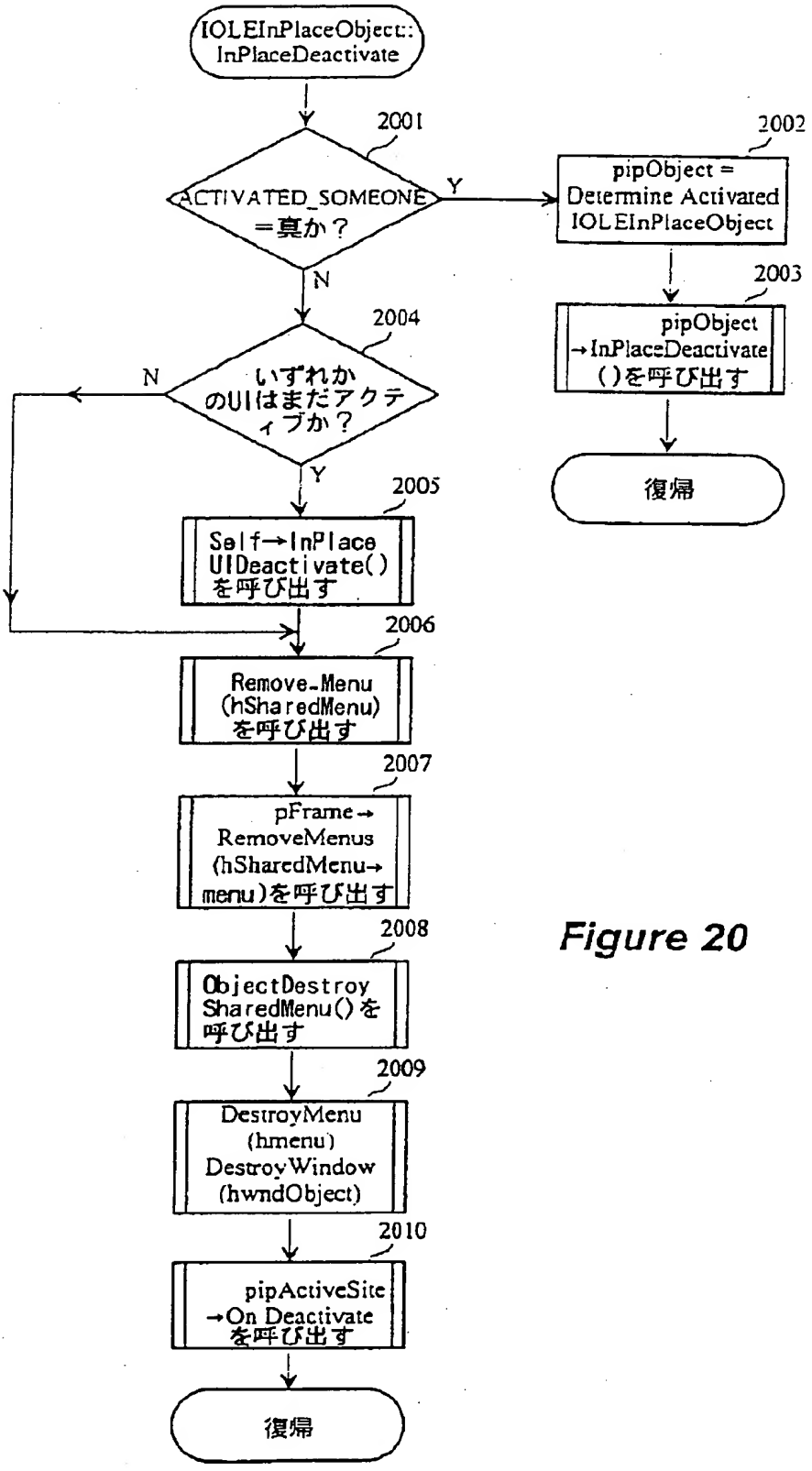

Figure 20

【図２１】

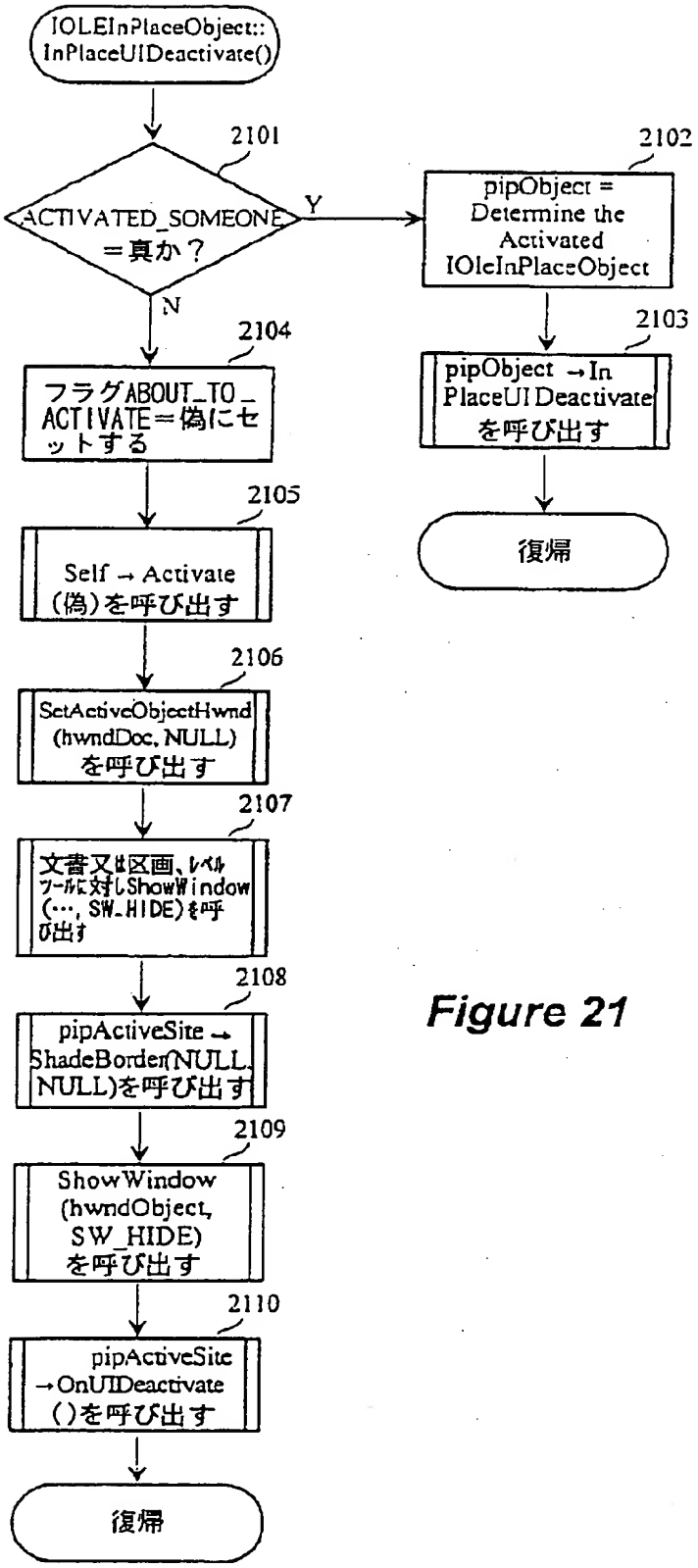

Figure 21

【図２２】

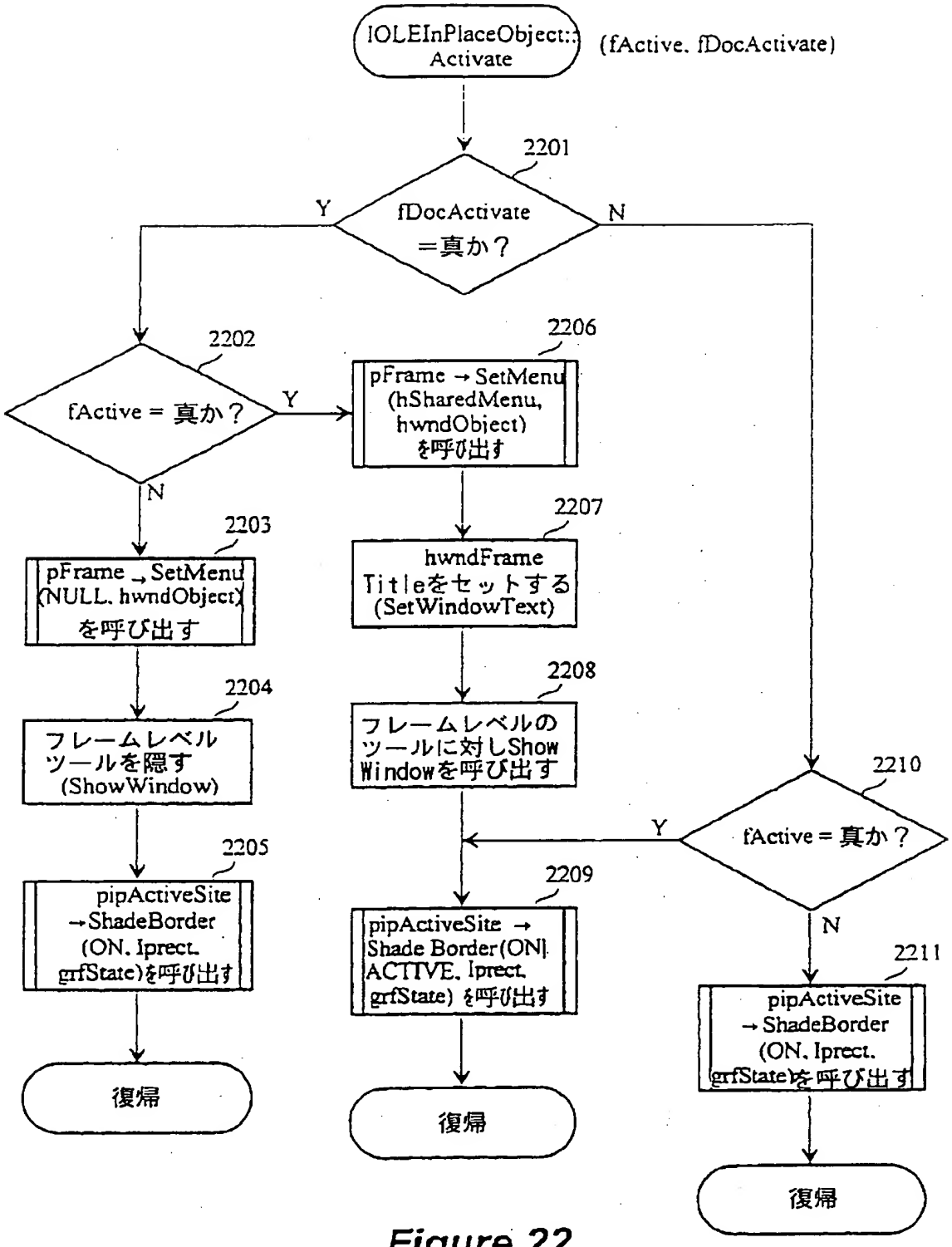

Figure 22

【図２３】

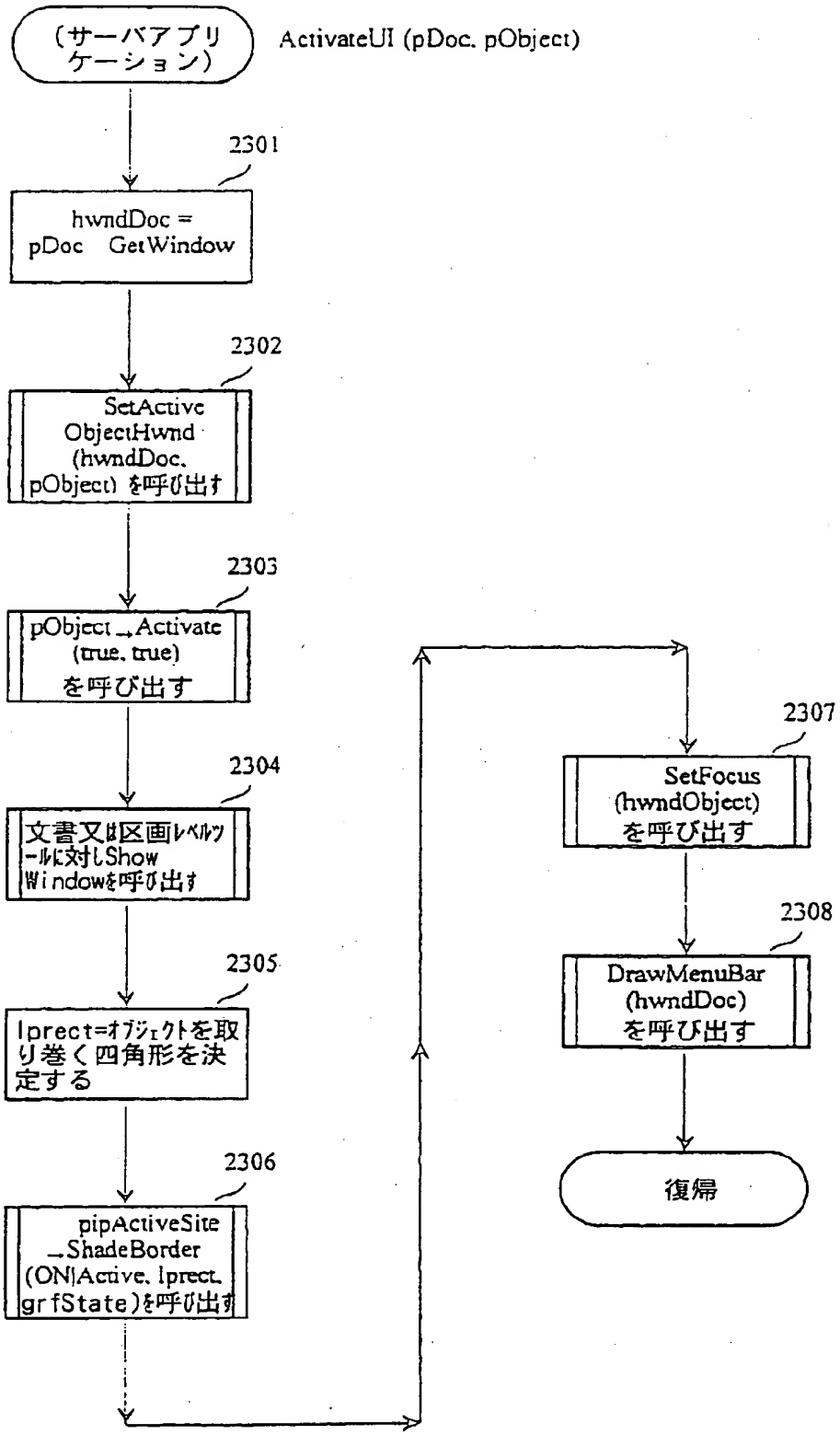

Figure 23

【図２４】

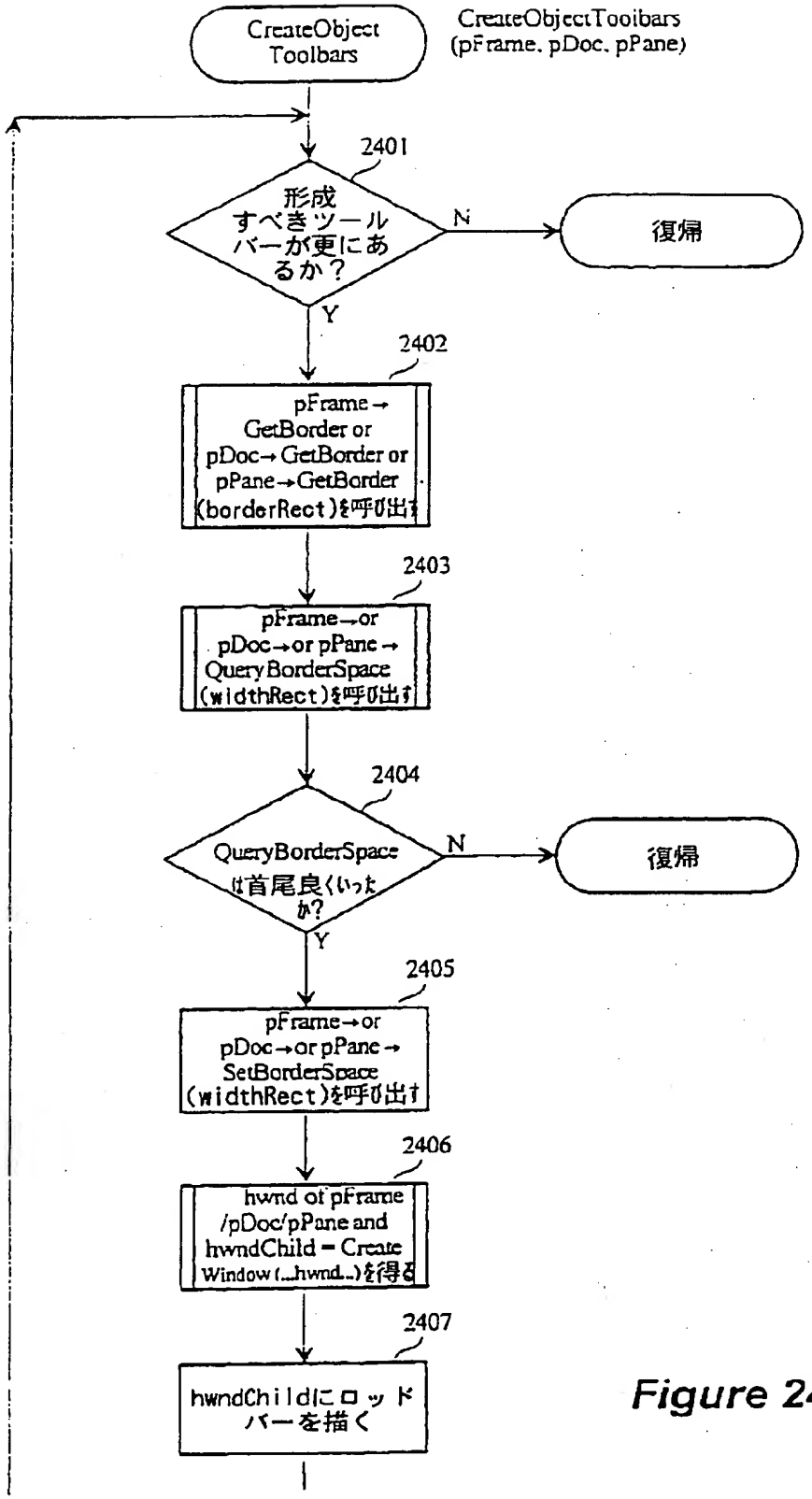

Figure 24

【図２５】

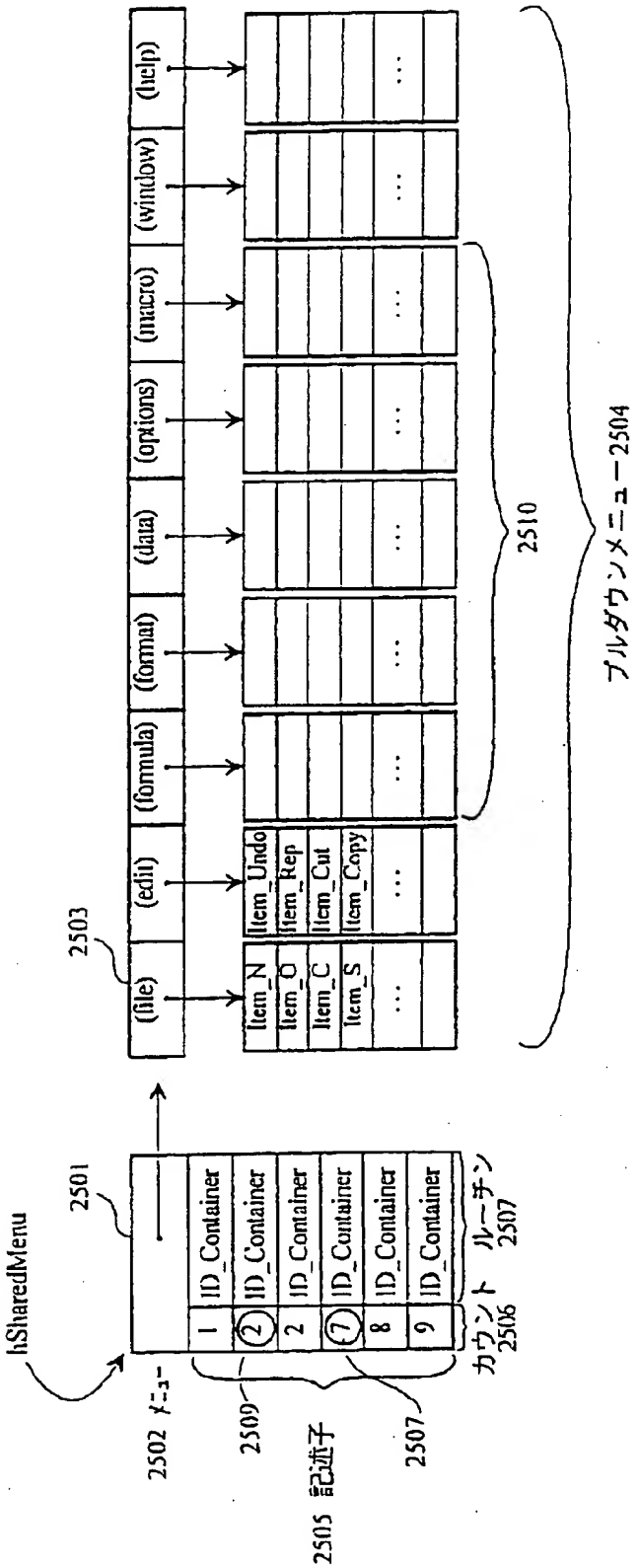

Figure 25

【図２６】

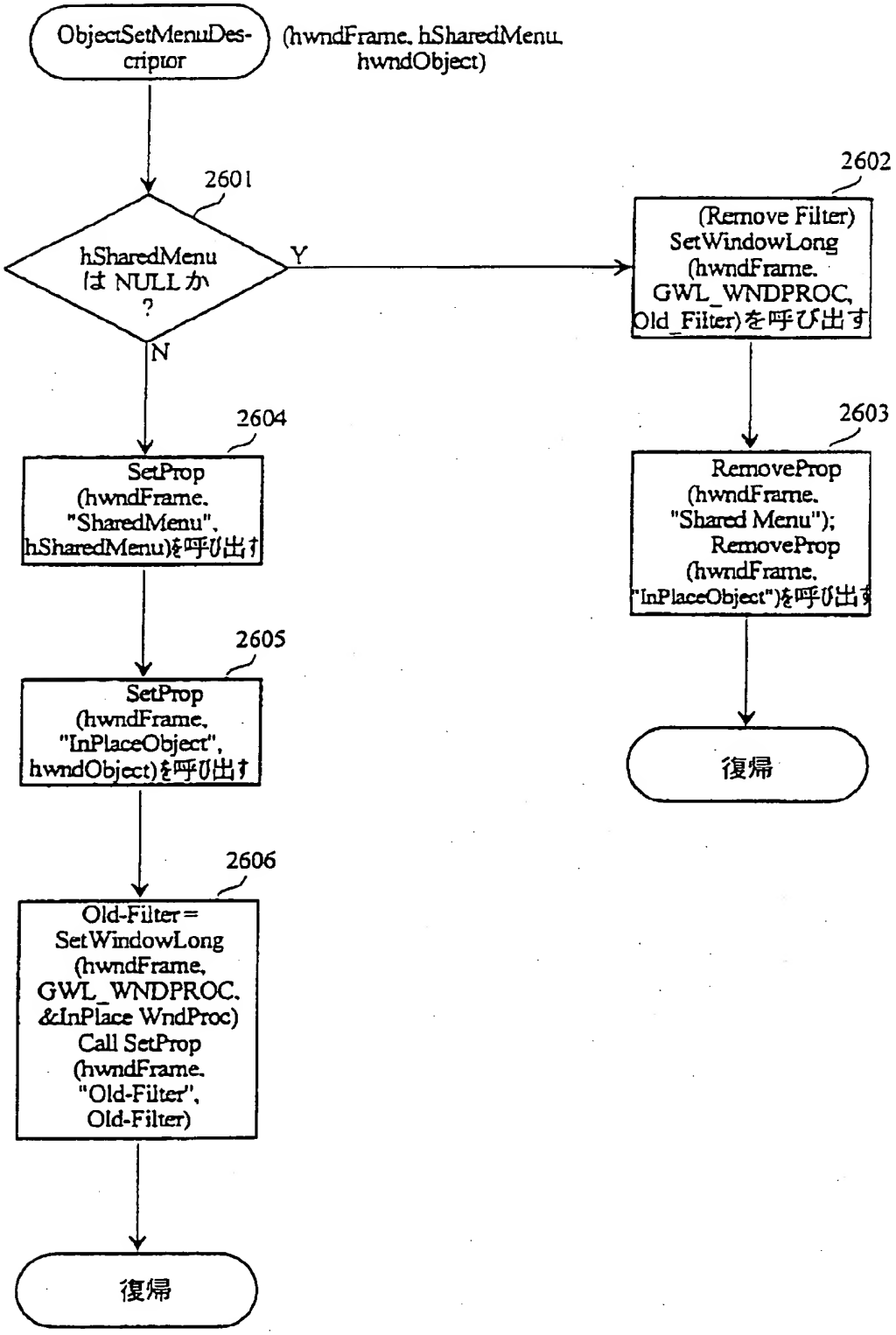

Figure 26

【図２７】

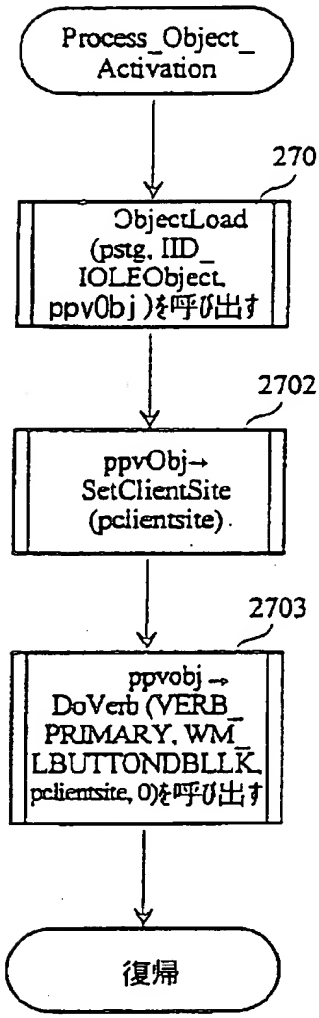

Figure 27

【図２８】

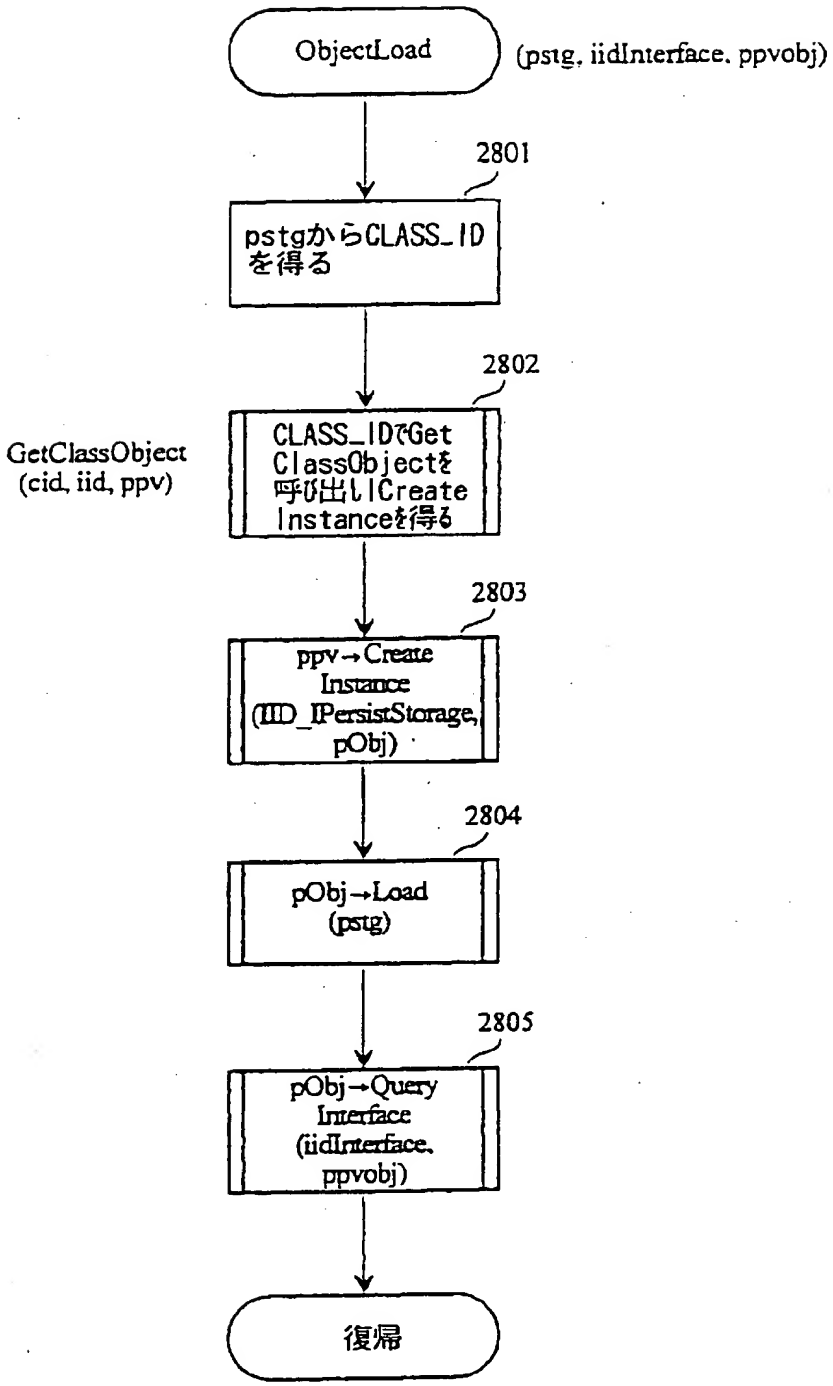

Figure 28

【図２９Ａ】

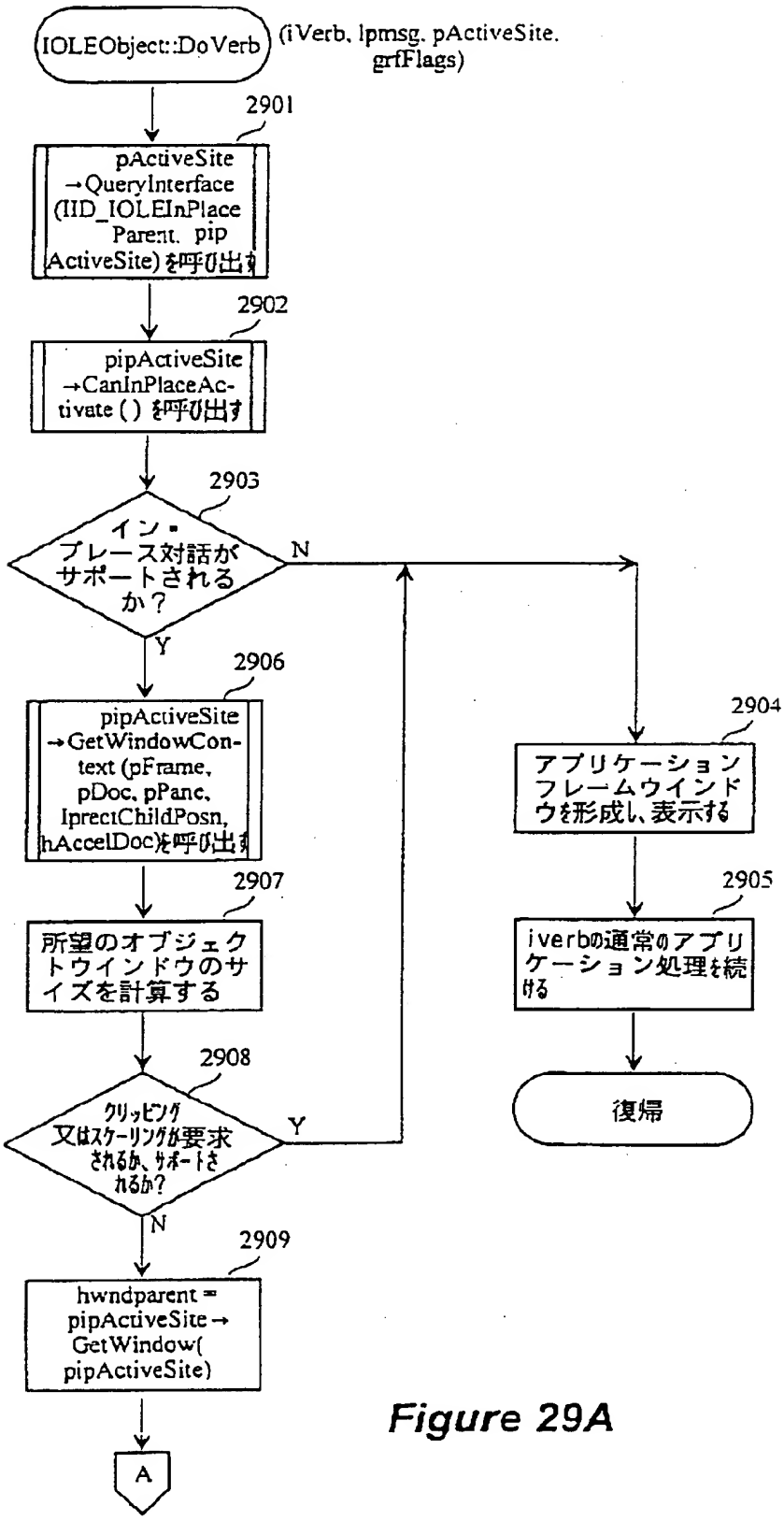

Figure 29A

【図２９Ｂ】

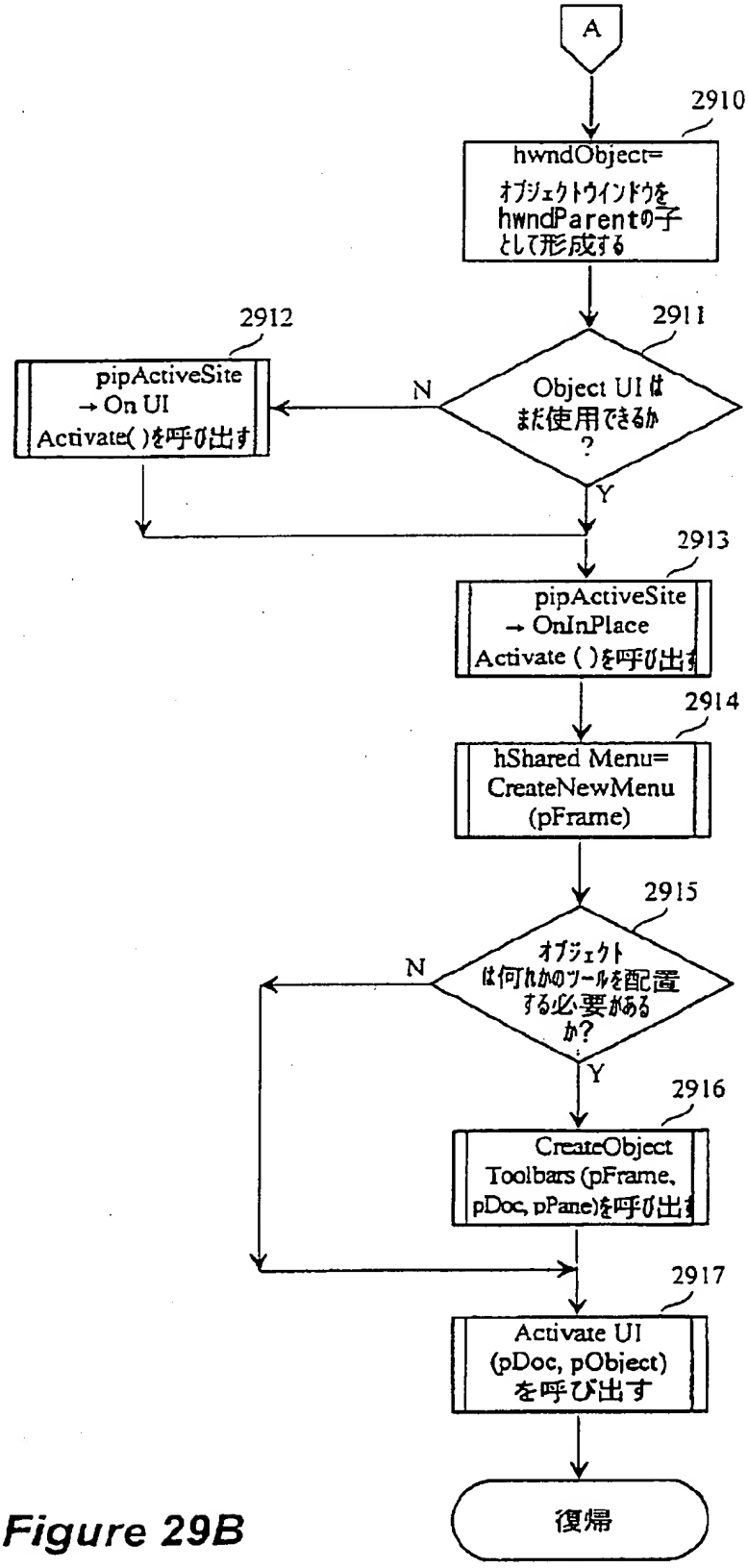

Figure 29B

【図３０】

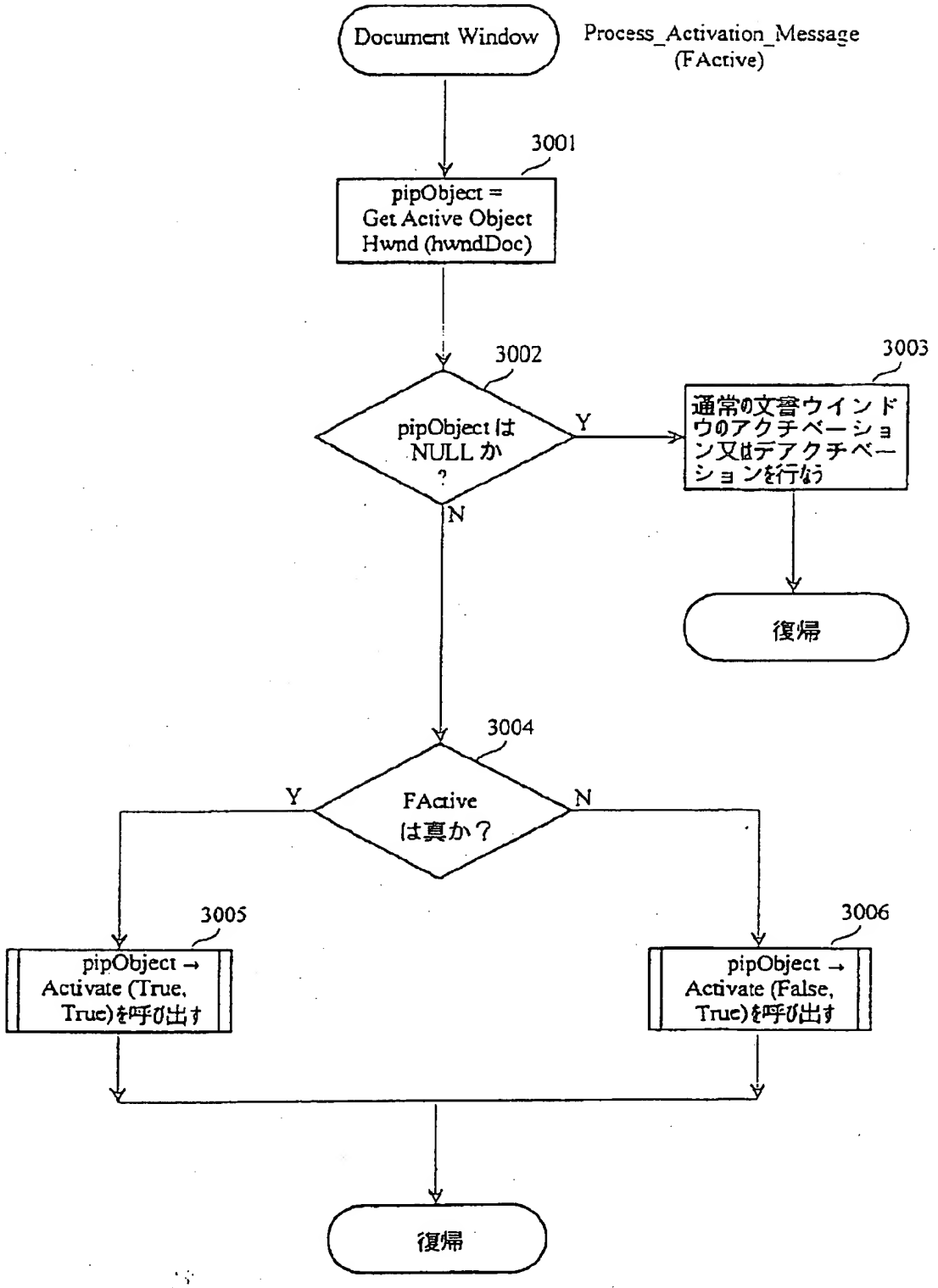

Figure 30

【図３１】

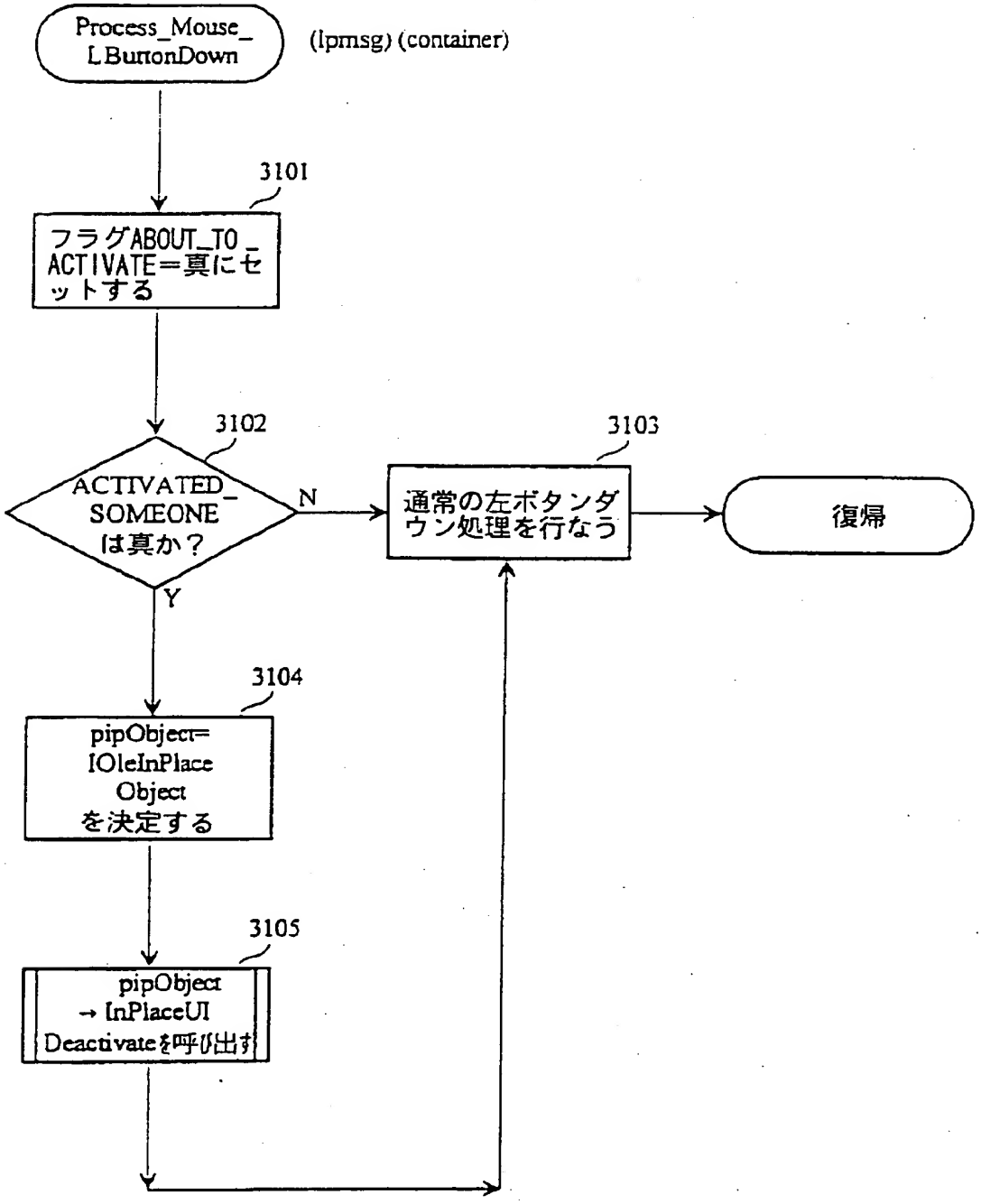

Figure 31

【手続補正書】

【提出日】１９９５年６月６日

【補正内容】

請求の範囲

１．コンピュータシステムにおいてコンテナーオブジェクト内に収容されたコンテニーオブジェクトをアクチベートするための方法であって、上記コンピュータシステムは、コードをスケジュールして実行するためのオペレーティングシステムを有し、上記コンテナーオブジェクトは、コンテナーウインドウ環境を伴うコンテナーアプリケーションコードを有し、上記コンテナーウインドウ環境は、コンテナーオブジェクトと対話するためのコンテナーリソースを有し、上記コンテニーオブジェクトは、該コンテニーオブジェクトと対話するためのサーバリソースを伴うサーバアプリケーションコードを有し、上記方法は、

上記コンテナーアプリケーションコードを実行し、コンテナーアプリケーションコードは、オペレーティングシステムによって個々にスケジュールすることができ、

上記コンテナーウインドウ環境を表示し、

上記表示されたコンテナーウインドウ環境内に上記コンテニーオブジェクトを表示し、

上記コンテニーオブジェクトをアクチベートし、そして

複数のサーバリソースを上記表示されたコンテナーウインドウ環境内で一体化して表示し、上記サーバリソースは、サーバアプリケーションコードの一部分であり、サーバアプリケーションコードは、オペレーティングシステムにより個々にスケジュールすることができて、上記一体化されたサーバリソースの中からあるサーバリソースをユーザが選択するときに、サーバアプリケーションが個々にスケジュールされたエンティティとして実行されそしてサーバリソースの選択を処理する、

という段階を備えたことを特徴とする方法。

２．上記コンテナーアプリケーションコードは、このコンテナーアプリケーションコードによって割り当てられて管理されるコンテナーメニューを有し、上記サ

ーバアプリケーションコードは、このサーバアプリケーションコードによって割り当てられて管理されるサーバメニューを有し、複数のサーバリソースを一体化して表示する上記段階は、

複合メニューバーを発生し、そして

その発生された複合メニューバーにサーバメニュー及びコンテナーメニューを記憶する、

というサブ段階を備えた請求項１に記載の方法。

３．上記記憶段階は、サーバメニュー及びコンテナーメニューを複合メニューバーにおいてインターリーブする請求項２に記載の方法。

４．上記コンテナーウインドウは、複数のメニューを表示するためのメニューバーを有し、そして上記複合メニューバーは、コンテナーアプリケーションのメニューバーとして表示される請求項２に記載の方法。

５．上記一体化されたサーバリソースの中からあるサーバリソースを選択するのに応答して、サーバアプリケーションコードを個々にスケジュールされたエンティティとして呼び出し、サーバリソースの選択を処理し、そして

上記表示されたコンテナーウインドウ環境からあるコンテナーリソースを選択するのに応答して、コンテナーアプリケーションコードを呼び出し、コンテナーリソースの選択を処理する、

という段階を更に備えた請求項１に記載の方法。

６．上記コンテナーアプリケーションコードは、データを表示するためのウインドウを上記表示されたコンテナーウインドウ環境内に有し、そしてそのウインドウ内にサーバリソースを配置するようにコンテナーアプリケーションコードと交渉する段階を備えた請求項１に記載の方法。

７．コンテナーウインドウ環境内に複数のサーバリソースを一体化して表示する上記段階は、その表示を行う前又は行うときに、どのサーバリソースを表示すべきかをコンテナーアプリケーションコードが知ることなく、行われる請求項１に記載の方法。

８．上記コンテナーアプリケーションコード及びサーバアプリケーションコード

は、単一のコンピュータプロセスにおいて個別のスレッドとして実行される請求項１に記載の方法。

９．複数のサーバリソースを一体化した後に、上記コンテナーウインドウ環境内のコンテニーオブジェクトをハイライト処理して表示し、サーバリソースがユーザ選択に使用できることを指示する段階を備えた請求項１に記載の方法。

10.上記コンテナーアプリケーションコードは、メッセージを受け取って処理するためのコンテナーメッセージハンドラーを有し、上記サーバアプリケーションは、メッセージを受け取って処理するためのサーバメッセージハンドラーを有し、上記コンテナーオブジェクトは、メッセージを受け取って処理するための現在メッセージハンドラーを有し、この現在メッセージハンドラー最初はコンテナーメッセージハンドラーにセットされ、そして複数のサーバリソースを一体化して表示する上記段階は、コンテナーオブジェクトの現在メッセージハンドラーを特殊なメッセージハンドラーにセットすることを含み、この特殊なメッセージハンドラーは、コンテナーリソースが選択されたときにコンテナーリソース選択メッセージをコンテナーメッセージハンドラーへ送ると共に、サーバリソースが選択されたときにサーバリソース選択メッセージをサーバメッセージハンドラーへ送るものである請求項１に記載の方法。

11.上記コンテナーウインドウ環境はウインドウを有し、上記サーバアプリケーションコードは、ウインドウを伴うサーバウインドウ環境を有し、そして更に、

上記サーバウインドウ環境のウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして指定し、

ユーザからメニューコマンドを受け取り、そして

上記メニューコマンドを受け取るのに応答して、

上記コンテナーウインドウ環境のウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして指定し、

メニューニューモニックを受け取り、そして

メニューニューモニックを受け取るのに応答して、サーバウインドウ環境のウインドウを、ユーザ入力を受け取るための入力フォーカスを有するものとして

再指定する、

という段階を含む請求項１に記載の方法。

12.上記コンテナーウインドウ環境のウインドウは、フレームウインドウである請求項１１に記載の方法。

13.上記コンピュータシステムはキー入力のためのキーボードを有し、上記コンテナーアプリケーションコードは、コンテナーリソースを選択するための複数のアクセラレーター一組合せを有し、上記サーバアプリケーションコードは、サーバリソースを選択するための複数のアクセラレーター組合せを有し、そして更に、上記コンテナーアプリケーションコードがアクセラレータキー組合せを受け取るときに、個別にスケジュールされたサーバアプリケーションコードを呼び出して、サーバリソースが選択されたかどうか決定する段階を備えた請求項１に記載の方法。

14.コンテニーオブジェクトをアクチベートする上記段階は、

コンテニーオブジェクトを選択し、そして

そのコンテニーオブジェクトに対して実行されるべきオペレーションを指定する、

という段階を更に備えた請求項１に記載の方法。

15.コンテナーオブジェクト内に収容されたコンテニーオブジェクトをアクチベートするためのコンピュータシステムであって、このコンピュータシステムは、コードをスケジュールして実行するオペレーティングシステムを有し、上記コンテナーオブジェクトは、コンテナーウインドウ環境を伴うコンテナーアプリケーションコードを有し、上記コンテナーウインドウ環境は、コンテナーオブジェクトと対話するためのコンテナーリソースを有し、上記コンテニーオブジェクトは、該コンテニーオブジェクトと対話するためのサーバリソースを伴うサーバアプリケーションコードを有するようなコンピュータシステムにおいて、

上記コンテナーウインドウ環境を表示する手段と、

上記表示されたコンテナーウインドウ環境内に上記コンテニーオブジェクトを表示する手段と、

上記表示されたコンテナーウインドウ環境内のコンテニーオブジェクトをアクチベートする手段と、

複数のサーバリソースを上記表示されたコンテナーウインドウ環境内で一体化して表示する手段と、

上記一体化され表示されたサーバリソースの中からあるサーバリソースが選択されたときにサーバアプリケーションを個々にスケジュールされたエンティ

ティとして実行しそしてサーバリソースの選択を処理する手段と、

を備えたことを特徴とするコンピュータシステム。

16.上記コンテナーアプリケーションコードは、このコンテナーアプリケーションコードにより割り当てられて管理されるコンテナーメニューを有し、上記サーバアプリケーションコードは、このサーバアプリケーションコードによって割り当てられて管理されるサーバメニューを有し、上記一体化して表示する手段は、複合メニューバーを発生しそしてその複合メニューバー内にサーバメニュー及びコンテナーメニューを記憶する手段を含む請求項１５に記載のシステム。

17.上記発生して記憶する手段は、サーバメニュー及びコンテナーメニューを複合メニューバーにおいてインターリーブする手段を含む請求項１６に記載のシステム。

18.上記コンテナーウインドウ環境は、複数のメニューを表示するメニューバーを有し、そして上記複合メニューバーは、コンテナーウインドウ環境のメニューバーとして表示される請求項１５に記載のシステム。

19.上記表示されたコンテナーウインドウ環境からコンテナーリソースが選択されたときにコンテナーリソース選択を処理するようにコンテナーアプリケーションを実行する手段を更に備えた請求項１５に記載のシステム。

20.上記コンテナーアプリケーションコードは、データを表示するためのウインドウをコンテナーウインドウ環境内に有し、そして上記一体化して表示する手段は、そのウインドウ内にサーバリソースを配置するようにコンテナーアプリケーションコードと交渉する手段を含む請求項１５に記載のシステム。

21.コンテナーウインドウ環境内に複数のサーバリソースを一体化して表示する

上記手段は、その表示を行う前又は行うときに、どのサーバリソースを表示すべきかをコンテナーアプリケーションコードが知ることなく、行われる請求項１５に記載のシステム。

22.上記コンテナーアプリケーションコード及びサーバアプリケーションコードは、単一のコンピュータプロセスにおいて個別のスレッドとして実行される請求項１５に記載のシステム。

23.上記コンテナーウインドウ環境内のコンテニーオブジェクトをハイライト処理して表示し、サーバリソースがユーザ選択に使用できることを指示するための手段を更に備えた請求項１５に記載のシステム。

24.上記コンピュータシステムはキー入力のためのキーボードを有し、上記コンテナーアプリケーションコードは、コンテナーリソースを選択するための複数のアクセラレータキー組合せを有し、上記サーバアプリケーションコードは、サーバリソースを選択するための複数のアクセラレータキー組合せを有し、そして更に、アクセラレータキー組合せを受け取るときにサーバリソースが選択されたかどうかを決定するためにサーバアプリケーションコードを個別に実行されるエンティティとして呼び出す手段を備えた請求項１５に記載のシステム。

25.上記アクチベートする手段は、コンテニーオブジェクトを選択しそしてそのコンテニーオブジェクトに対して実行されるべきオペレーションを指定する請求項１５に記載のシステム。

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Intc. onal Application No
PCT/US 93/11468

A. CLASSIFICATION OF SUBJECT MATTER
IPC 5 G06F9/44 G06F3/033

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 5 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	BYTE, vol.17, no.14, December 1992, LONDON, GB pages 153 - 160 PETER WAYNER: 'Brave New Desktop' see page 155, right column, line 8 - page 156, middle column, line 2; figures ---	1-16, 22-34
A	'Microsoft Windows Version 3.1 User's Guide' , MICROSOFT CORPORATION , US see page 487, paragraph 1 - page 489, last paragraph ---	1-16, 22-34
A	EP,A,0 215 203 (IBM) 25 March 1987 see the whole document ---	1-16, 22-34

-/--

[X] Further documents are listed in the continuation of box C. [X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
7 April 1994	29. 08. 94

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax (+ 31-70) 340-3016	Fonderson, A

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Int. ional Application No
PCT/US 93/11468

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IBM TECHNICAL DISCLOSURE BULLETIN, vol.27, no.10B, March 1985, NEW YORK, US page 5998 'Object-Specific Command Bar' see the whole document	1-7, 14-16, 22-28
A	EP,A,0 304 071 (WANG LABORATORIES INC.) 22 February 1989 see abstract	1-16, 22-34

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/11468

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

SEE ANNEXED SHEET

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searches without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

1-16, 22-34

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

INVITATION TO PAY ADDITIONAL FEES

International application No.
PCT/ US 93/ 11468

1. Claims 1-16, 22-34 : Method and system of interacting with a contained object within the context of its containing application.

2. Claim 17 : Method for dynamically combining window hierarchies in a computer system.

3. Claims 18-20 : Method for scrolling a window containing another window which contains selected data.

4. Claim 21 : Method for visually emphasising a selected object which is displayed on a display device.

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inte ional Application No
PCT/US 93/11468

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-0215203	25-03-87	US-A-	4815029	21-03-89
		JP-A-	62072059	02-04-87
EP-A-0304071	22-02-89	AU-B-	607795	14-03-91
		AU-A-	2094388	23-02-89
		JP-A-	1126736	18-05-89
		US-A-	5206951	27-04-93
		US-A-	5261080	09-11-93
		US-A-	5303379	12-04-94
		US-A-	5226161	06-07-93

フロントページの続き

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		
		9288－5Ｌ	Ｇ０６Ｆ	15/20	５８０　Ｊ

(72)発明者　マッキチャン　バリー　ビー
アメリカ合衆国　ワシントン州　98110
ベインブリッジ　アイランド　マンザニタ
ロード　ノースイースト　12730
(72)発明者　マックダニエル　リチャード
アメリカ合衆国　ペンシルバニア州
15206　ピッツバーグ　スタントン　アベ
ニュー　6017　アパートメント　1
(72)発明者　レマラ　ラオ　ヴィー
アメリカ合衆国　ワシントン州　98072
ウッディンヴィル　ノースイースト　ワン
ハンドレッドアンドフィフティファースト
ストリート　19011
(72)発明者　ウィリアムズ　アントニー　エス
アメリカ合衆国　ワシントン州　98053
レッドモンド　ノースイースト　フォーテ
ィシックスス　ストリート　22542

【要約の続き】
ス選択を処理する。